

Elections with Few Voters: Candidate Control Can Be Easy*

Jiehua Chen^{†1}, Piotr Faliszewski^{‡2}, Rolf Niedermeier^{§1}, and Nimrod Talmon^{¶1}

¹Institut für Softwaretechnik und Theoretische Informatik, TU Berlin,
Germany

²AGH University of Science and Technology, Krakow, Poland

Abstract

We study the computational complexity of candidate control in elections with few voters (that is, we take the number of voters as a parameter). We consider both the standard scenario of adding and deleting candidates, where one asks if a given candidate can become a winner (or, in the destructive case, can be precluded from winning) by adding/deleting some candidates, and a combinatorial scenario where adding/deleting a candidate automatically means adding/deleting a whole group of candidates. Our results show that the parameterized complexity of candidate control (with the number of voters as the parameter) is much more varied than in the setting with many voters.

1 Introduction

Election control problems model the issue of affecting the result of an election by either introducing some new candidates/voters or by removing some of them from the election. We study the complexity of election control by adding and deleting candidates, for the case where the election involves a few voters only. We focus on very simple, practical voting rules such as Plurality, Veto, and t -Approval, but we also discuss some more involved ones. To analyze the effect of a small number of voters, we use the formal tools of parameterized complexity theory.

From the point of view of classical complexity theory, candidate control is NP-hard for almost all typically studied voting rules (even for the Plurality rule; though some natural examples of polynomial-time candidate control problems exist as well). It turns out that for the case of elections with few voters (i.e., for control problems parameterized by the number of

*PF was supported by the DFG project PAWS (NI 369/10) and by AGH University grant 11.11.230.124 (statutory research). NT was supported by the DFG Research Training Group “Methods for Discrete Structures” (GRK 1408). This work has been partly supported by COST Action IC1205 on Computational Social Choice. To appear in the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15).

[†]jiehua.chen@tu-berlin.de

[‡]faliszew@agh.edu.pl

[§]rolf.niedermeier@tu-berlin.de

[¶]nimrodtalmon77@gmail.com

(a) Approval-based voting rules

Problem	Plurality	Veto	t -Approval	t -Veto
\mathcal{R} -CCAC	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP
\mathcal{R} -CCDC	FPT	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP
\mathcal{R} -DCAC	FPT	FPT	FPT	FPT
\mathcal{R} -DCDC	FPT	FPT	FPT	FPT
\mathcal{R} -COMB-CCAC	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP
\mathcal{R} -COMB-CCDC	para-NP-h (1)	para-NP-h (1)	para-NP-h (1)	para-NP-h (1)
\mathcal{R} -COMB-DCAC	FPT	FPT	W[1]-h / XP	? / XP
\mathcal{R} -COMB-DCDC	para-NP-h (3)	para-NP-h (1)	para-NP-h (2)	para-NP-h (1)

(b) Other voting rules

Problem	Borda	Copeland ^{α}	Maximin
\mathcal{R} -CCAC	para-NP-h (10)	para-NP-h (20) \spadesuit	para-NP-h (10)
\mathcal{R} -CCDC	para-NP-h (10)	para-NP-h (26) \spadesuit	P \clubsuit
\mathcal{R} -DCAC	P \diamond	P \clubsuit	
\mathcal{R} -DCDC	P \diamond	P \clubsuit	
\mathcal{R} -COMB-CCAC	para-NP-h (2)	para-NP-h (3) \spadesuit	para-NP-h (6)
\mathcal{R} -COMB-CCDC	para-NP-h (1)	para-NP-h (1) \spadesuit	para-NP-h (1)
\mathcal{R} -COMB-DCAC	para-NP-h (2)	para-NP-h (3)	P
\mathcal{R} -COMB-DCDC	para-NP-h (2)	para-NP-h (3)	para-NP-h (5)

Table 1: The complexity of candidate control (constructive (CC) and destructive (DC), adding candidates (AC) and deleting candidates (DC)) problems for varying voting rules \mathcal{R} parameterized by the number of voters (for t -Approval and t -Veto we mean $t \geq 2$; for Copeland ^{α} , we mean $0 \leq \alpha \leq 1$; notice that the results by Betzler and Uhlmann [2] hold only for $\alpha \in \{0, 1\}$). Results marked with \clubsuit and \diamond are due to Faliszewski et al. [16, 14], those marked with \heartsuit are due to Loreggia et al. [24], and those marked with \spadesuit follow from the work of Betzler and Uhlmann for $\alpha \in \{0, 1\}$ and are due to this paper for the remaining values. Cells containing statements of the form “para-NP-h (z)” mean that the relevant problem is NP-hard even with only z voters. Question mark (?) means that the exact complexity is still open.

voters), the landscape of the complexity of candidate control is quite varied and, indeed, sometimes quite surprising (see Table 1 for an overview of our results). In addition to the standard candidate control problems, we also study their *combinatorial* variants, where it is possible to add or delete whole groups of candidates at unit cost. In this we follow the path initiated by Chen et al. [9], who introduced combinatorial voter control.

Motivation. There is a number of settings in which it is most natural to consider elections with few voters (and, typically, many candidates). Let us look at several examples.

Hiring committee. Consider a university department which is going to hire a new faculty member. Typically the committee consists of relatively few faculty members, but it may consider hundreds of applications for a position.

Holiday planning. Consider a group of people who are planning to spend holidays together. The group typically would consist of no more than a dozen persons, but—technically—they have to choose from all the possible options provided by the travel offices, hotels,

airlines, etc. This example is particularly relevant to the case of multi-agent systems: One may foresee that in the future we will delegate the task of finding the most satisfying holiday location to our personal software agents that will negotiate with travel offices and other travelers on our behalf.

Meta-search engine. Dwork et al. [12] argued that one can build a web meta-search engine that queries several other search engines (the few voters) regarding a given query, aggregates their rankings of the web pages (the many candidates), and outputs the consensus ranking.

In all these examples, it is clear that before we actually hold an election, the voters (or, some particular individual) first shrink the set of candidates. In the case of the hiring committee, most of the applications are removed from the considerations early in the evaluation process. The people planning holidays first, implicitly, remove most of possible holiday options and, then, remove those candidates that do not fit their preferences completely (e.g., too expensive offers). The search engines usually disregard those web pages that appear completely irrelevant to a given query.

This natural process of modifying the candidate set, however, creates a natural opportunity for manipulating the result. A particularly crafty agent may remove those candidates that prevent his or her favorite one from winning. Similarly, after the initial process of thinning down the candidate set, a voter may request that some candidates are added back into consideration, possibly to help his or her favorite candidate. More importantly, it is quite realistic to assume that the voters in a small group know each other so well as to reliably predict each others' votes (this is particularly applicable to the example of the hiring committee). Thus, we believe that it is natural and relevant to study the complexity of candidate control parameterized by the number of voters. While control problems do not model the full game-theoretic process of adding/deleting candidates, they allow agents to compute what effects they might be able to achieve.

Finally, it is quite natural to consider the case where deleting (adding) a particular candidate means also deleting (adding) a number of other ones. For example, if a hiring committee removes some candidate from consideration, it might have to also remove all those with weaker publication records; if people planning holidays disregard some expensive hotel, they might also want to remove those that cost more. Thus, we also study *combinatorial variants* of candidate control problems that model such settings.

Main contributions. Our research has shown some surprising patterns that were not (nearly as) visible in the context of classical complexity analysis of election control:

1. (Non-combinatorial) destructive candidate control is easy for all our voting rules, either in the fixed-parameter tractability sense or via outright polynomial-time algorithms.
2. In the combinatorial setting, control by deleting candidates appears to be computationally harder than control by adding candidates.

We also found an interesting difference in the complexity of non-combinatorial constructive control by deleting candidates between Plurality and Veto rules (this is especially interesting since there is no such difference for the adding candidates case).

Our results (see Table 1; formal definitions follow in the next section) are of four types (with the exception of *t*-Veto-Comb-DCAC which is only in XP): for each of our problems we show that it either is in P, is in FPT, is W[1]-hard but has an XP-algorithm, or is para-NP-hard (in each case the parameter is the number of voters). Naturally, the first type of

results is the most positive¹ (unconditionally efficient algorithms) and the second type is quite positive too (the exponential part of the running time of an algorithm depends only on the number of voters). The third kind is less positive (W[1]-hardness precludes existence of FPT algorithms, but membership in XP means that there are algorithms that are polynomial-time if the number of voters is a constant). The last kind is the most negative (NP-hardness even for a constant number of voters; this precludes membership in XP).² We introduce several new proof techniques to establish our results. For clarity of reading, we only sketch some of our proofs in the main text. Complete formal proofs are given in the Appendix.

Related Work. The complexity study of election control was introduced by Bartholdi et al. [1], who were later followed by numerous researchers, including, e.g., Hemaspaandra et al. [18], Meir et al. [26], and many others (we point the reader to the survey by Faliszewski et al. [15] and to several recent papers on the topic Parkes and Xia [29], Erdélyi et al. [13], Rothe and Schend [30]). Briefly put, it turns out that for standard voting rules, control problems are typically NP-hard.

There is a growing body of research regarding the parameterized complexity of voting problems (see, e.g., the survey by Betzler et al. [4]), where typical parameters include the solution size (e.g., the number of candidates that can be added) and the election size (i.e., the number of candidates or the number of voters). For the solution size as the parameter, control problems usually turn out to be hard Betzler and Uhlmann [2], Liu et al. [23], Liu and Zhu [22]. On the contrary, taking the number of candidates as the parameter almost always leads to FPT (fixed-parameter tractability) results (see, e.g., the papers by Faliszewski et al. [16] and by Hemaspaandra et al. [19]). However, so far, only Betzler and Uhlmann [2] considered a *control* problem parameterized by the number of voters (for the Copeland rule), and Brandt et al. [7] showed NP-hardness results of several winner determination problems even for constant number of voters. The parameter “number of voters” also received some limited attention in other voting settings (Betzler et al. [3]; Dorn and Schlotter [10] Dorn and Schlotter [10]; Bredereck et al. [8] Bredereck et al. [8]).

The study of combinatorial control was recently initiated by Chen et al. [9], who focused on voter control. We stress that our combinatorial view of control is different from the studies of combinatorial voting domains Boutilier et al. [6], Xia and Conitzer [32], Mattei et al. [25].

2 Preliminaries

Elections. An election $E = (C, V)$ consists of a set of candidates $C = \{c_1, \dots, c_m\}$ and a collection $V = (v_1, \dots, v_n)$ of voters. Each voter v_ℓ has a preference order (vote), often denoted \succ_ℓ , which ranks the candidates from the one that v_ℓ likes most to the one that v_ℓ likes least. For example, if $C = \{c_1, c_2, c_3\}$ then a voter with preference order $c_1 \succ c_2 \succ c_3$ would most like c_1 to be a winner, then c_2 , and then c_3 . For a voter v_ℓ and two candidates c_i, c_j , we sometimes write $v_\ell: c_i \succ c_j$ to indicate that v_ℓ prefers c_i to c_j . If A is some subset of candidates, then writing A within a preference order description (e.g., $A \succ a \succ b$, where a and b are some candidates) means listing members of A in some arbitrary, but fixed, order. Writing \overleftarrow{A} means listing the candidates in the reverse of this order. Given an election $E = (C, V)$, for each two candidates $c_i, c_j \in C$, we define $N_E(c_i, c_j) := \|\{v_\ell \mid v_\ell: c_i \succ c_j\}\|$.

¹We note that we evaluate the results from the computational complexity perspective and, hence, regard computational efficiency as positive.

²Naturally, we use the standard complexity-theoretic assumptions that $P \neq NP$ and $FPT \neq W[1]$.

A voting rule \mathcal{R} is a function that given an election $E = (C, V)$ outputs a set $\mathcal{R}(E) \subseteq C$ of candidates that tie as winners (i.e., we use the non-unique-winner model, where the candidates in $\mathcal{R}(E)$ are equally successful). We study the following standard voting rules (in each case, the candidates who receive the highest number of points are the winners):

t-Approval and t-Veto. Under *t-Approval* (where $t \geq 1$ is an integer), each candidate gets a point for each voter that ranks him or her among the top t positions. For m candidates, *t-Veto* is a nickname for $(m - t)$ -Approval (we often view the score of a candidate under *t-Veto* as the number of vetoes, i.e., the number of times he or she is ranked among bottom t positions). We refer to 1-Approval and 1-Veto as the Plurality rule and the Veto rule, respectively, and we jointly refer to the voting rules in this group as approval-based rules.

Borda rule and Maximin rule. Under the Borda rule, in election $E = (C, V)$ each candidate $c \in C$ receives $\sum_{d \in C \setminus \{c\}} N_E(c, d)$ points. (It is also convenient to think that Borda, for each voter v , gives each candidate c as many points as the number of candidates that v ranks c ahead of.) Under Maximin, each candidate $c \in C$ receives $\min_{d \in C \setminus \{c\}} N_E(c, d)$ points.

Copeland $^\alpha$ rule. Under the Copeland $^\alpha$ rule (where α is rational, $0 \leq \alpha \leq 1$), in election $E = (C, V)$ each candidate c receives $\|\{d \in C \setminus \{c\} \mid N_E(c, d) > N_E(d, c)\}\| + \alpha \|\{d \in C \setminus \{c\} \mid N_E(c, d) = N_E(d, c)\}\|$ points.

Control Problems. We study *candidate control* in elections, considering both constructive control (CC) and destructive control (DC), by either adding candidates (AC) or deleting candidates (DC). Following the work by Chen et al. [9], we also consider combinatorial variants of our problems, where adding/deleting a single candidate automatically adds/deletes a whole group of other candidates. In these *combinatorial variants* (denoted with a prefix Comb), we use bundling functions κ such that for each candidate c , $\kappa(c)$ is a set of candidates that are also added if c is added (or, that are also deleted if c is deleted). For each candidate c , we require that $c \in \kappa(c)$ and call $\kappa(c)$ the bundle of c .³ If B is some subset of candidates, by $\kappa(B)$ we mean $\bigcup_{c \in B} \kappa(c)$. Bundling functions are encoded by explicitly listing their values for all the arguments. Formally, given a voting rule \mathcal{R} , our problems are defined as follows.

\mathcal{R} -COMB-CCAC

Input: An election (C, V) , a set A of unregistered candidates such that the voters from V have preference orders over $C \cup A$, a preferred candidate $p \in C$, a bundling function κ , and a non-negative integer k .

Question: Is there a set $A' \subseteq A$ with $\|A'\| \leq k$ such that $p \in \mathcal{R}(C \cup \kappa(A'), V)$?

\mathcal{R} -COMB-CCDC

Input: An election (C, V) , a preferred candidate $p \in C$, a bundling function κ , and a non-negative integer k .

Question: Is there a set $C' \subseteq C$ with $\|C'\| \leq k$ such that $p \in \mathcal{R}(C \setminus C', V)$?

The destructive variants of our problems, \mathcal{R} -COMB-DCAC and \mathcal{R} -COMB-DCDC, are defined analogously except that we replace the preferred candidate p with the despised candidate d , and we ask if it is possible to ensure that d is *not* a winner of the election. In the DCDC case,

³Whenever we delete candidates from an election, these candidates are also implicitly deleted from the voters' preference orders.

we explicitly disallow deleting any bundle containing the despised candidate. In the standard, non-combinatorial, variants of control we omit the prefix “Comb” and assume that for each candidate c we have $\kappa(c) = \{c\}$, omitting the bundling function in discussions.

Our model of combinatorial candidate control is about the simplest that one can think of. Indeed, in a scenario with m candidates, there are at most m corresponding bundles of candidates that can be added/deleted. In real life, one might expect many more. However, on the one hand, even such a simple model turns out to be computationally difficult and, on the other hand, we believe that it is instructive to consider such a simplified model first. In many cases (e.g., combinatorial constructive control by deleting candidates) we already obtain very strong hardness results.

Parameterized Complexity. Many of our results regard hardness with respect to the hierarchy of parameterized intractability:

$$\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \dots \subseteq \text{XP}.$$

The classes $\text{W}[1]$ and $\text{W}[2]$ can be defined, for example, through their complete problems, **MULTI-COLORED CLIQUE** and **SET COVER**: $\text{W}[1]$ and $\text{W}[2]$ contain those problems that reduce to, respectively, **MULTI-COLORED CLIQUE** and **SET COVER** in the parameterized sense. A parameterized reduction from a parameterized problem L to a parameterized problem L' is a function that, given an instance (I, p) , computes in $f(p) \cdot |I|^{O(1)}$ time an instance (I', p') , such that $p' \leq g(p)$ and $(I, p) \in L \Leftrightarrow (I', p') \in L'$; indeed, in this paper all reductions can actually be performed in polynomial time.

Definition 1. An instance of **MULTI-COLORED CLIQUE** consists of a graph $G = (V(G), E(G))$ and a non-negative integer h . The vertex set $V(G)$ is partitioned into h sets, $V_1(G), \dots, V_h(G)$, each one-to-one corresponding to one out of h colors. We ask if there are h vertices v_1, \dots, v_h such that for each i , $1 \leq i \leq h$, $v_i \in V_i(G)$, and each pair is connected by an edge. We call the set of these h vertices a multi-colored clique of size h .

Definition 2. An instance of **SET COVER** consists of a ground set $X = \{x_1, \dots, x_n\}$, a family $\mathcal{S} = \{S_1, \dots, S_m\}$ of subsets of X , and a non-negative integer h (taken to be the parameter). We ask if it is possible to pick at most h sets from \mathcal{S} so that their union is X . We call the collection these h sets a set cover of size h .

We say that a problem is para-NP-hard if there is a proof of its NP-hardness that produces an instance in which the value of the parameter is bounded by a constant. If a problem is para-NP-hard for some parameter, then it even cannot belong to XP for this parameter (unless $\text{P} = \text{NP}$). The textbooks on parameterized complexity theory offer more information [11, 17, 28].

3 Overview of Proof Techniques

We introduce several proof techniques that can be useful in studying the complexity of election problems parameterized by the number voters. We use the following techniques (the first two are, perhaps, most interesting):

Multi-Colored Clique Technique. This is a technique used for establishing $\text{W}[1]$ -hardness results. The idea is to give a reduction from **MULTI-COLORED CLIQUE** (MCC) parameterized by the clique order (a variant of the standard **CLIQUE** problem, better suited for the parameterized complexity results, where each vertex has one of h colors and we

seek a clique of order h with each vertex of a different color): Given an MCC-instance, we introduce a candidate for each vertex and two candidates for each edge, and—in essence—we have to ensure that we add only the candidates (delete all but the candidates) that correspond to a multi-colored clique. We enforce this constraint using pairs of carefully crafted votes such that if we have two vertices but not an edge between them, then some candidate receives one more point than it should have for our preferred candidate to win. Note that the colors help to bound the number of voters needed for the construction. See Theorem 1 for a sample proof.

Cubic Vertex Cover Technique. This is a technique used for establishing para-NP-hardness results for non-combinatorial constructive candidate controls. The crucial idea of the technique is that the edges in a cubic graph can be partitioned into four disjoint matchings, which allows one to encode all the information regarding the graph in a constant number of votes, in a way that ensures that the actions of adding/deleting candidates correspond to covering edges. A sample proof is given in Theorem 5.

Set-Embedding Technique. This is a very simple technique for showing para-NP-hardness results for combinatorial control by adding/deleting candidates. The idea is to reduce from the standard SET COVER problem using the bundling function to encode sets. Due to the power of bundling, a constant number of voters suffices for the reduction. A sample proof is given for Theorem 2.

Signature Technique. This is a group of two very similar techniques for showing FPT results (usually for destructive control). The first technique in the group works for control by adding candidates problems and relies on the fact that often it is possible to limit the number of candidates that one has to consider by identifying their most crucial properties (such as the subsets of voters where the candidates are ranked ahead of some given candidate; we refer to these properties as signatures). The second technique applies to control by deleting candidates. A sample proof using the first technique is given in Theorem 1.

4 Approval-Based Rules

In this section, we consider t -Approval and t -Veto rules. These are perhaps the simplest and most frequently used rules, so results regarding them are of particular interest.

We start by looking at the Plurality rule and the Veto rule. In terms of standard complexity theory, control by adding/deleting candidates (constructive and destructive) is NP-complete for both of them (Bartholdi et al. [1]; Hemaspaandra et al. [18]). However, if we parameterize by the number of voters, the results change quite drastically. On the one hand, the results for analogous types of (non-combinatorial) control for these rules differ (for example, Plurality-CCDC is in FPT but Veto-CCDC is W[1]-hard; this is quite unexpected given the similarity and simplicity of Plurality and Veto), and, on the other hand, combinatorial and non-combinatorial control problems behave differently. For example, in combinatorial control, the *deleting* candidates case is para-NP-hard for all the rules, but the *adding* candidates case is either in FPT or W[1]-hard (but in XP).

Theorem 1. *When parameterized by the number of voters, (1) for Plurality and Veto, DCAC and DCDC are both in FPT, (2) Plurality-CCAC and Veto-CCAC are both W[1]-hard, and (3) Plurality-CCDC is in FPT, while Veto-CCDC is W[1]-hard.*

Proof sketch for Plurality-DCAC. First, we guess a candidate p which is to defeat the despised candidate d (such a candidate must exist in a “yes”-instance; if p is an unregistered candidate, then we add it and decrease k by one).

Let $m := \|A\| + \|C\|$ be the total number of candidates and n be the number of voters. For each unregistered candidate a , we define its signature to be the collection of votes restricted to candidates p , d , and a , with each occurrence of a replaced by a global symbol x . Adding a single candidate with a given signature has the same effect on the score difference of d and p as adding several candidates with the same signature. Thus, we partition the set of unregistered candidates into equivalence classes based on their signatures, and, for each signature, remove all unregistered candidates but one. We also remove all the registered candidates that do not score any points in the original election. Altogether, we are left with at most n registered candidates and at most 3^n unregistered ones (the maximum number of different signatures). We solve this instance by brute-forcing all at-most- k -sized subsets of the unregistered candidates. This gives running time of the form $O(3^{n^2} \cdot \text{poly}(m, n))$ since $k \leq n$. Finally, we remark that by using exponential space we can design a more complicated $O(2^n \cdot m \cdot n)$ -time algorithm for Plurality-DCAC. \square

Proof sketch for Plurality-CCAC. We give a reduction from the W[1]-hard problem MULTI-COLORED CLIQUE parameterized by the clique order. In this problem, we are given an undirected graph $G = (V(G), E(G))$ whose vertices are partitioned into exactly h disjoint sets, $V_1(G), \dots, V_h(G)$ such that for each i , $V_i(G)$ consists of exactly n' vertices with color i . We ask if there is an order- h clique containing a vertex for each color. We rename the vertices so that for each i , $1 \leq i \leq h$, we have $V_i(G) = \{v_1^{(i)}, \dots, v_{n'}^{(i)}\}$. W.l.o.g., we assume that G has edges between vertices of different colors only.

We construct a Plurality-CCAC instance as follows: The registered candidates are p (the preferred one) and d . We have one unregistered candidate v for each vertex v , and two unregistered candidates, (u, v) , (v, u) , for each edge $\{u, v\}$.

To describe the votes, we need the following notation. Let i and j be two distinct colors. Let $E(i, j)$ denote the set of all edge candidates (u, v) , where $u \in V_i(G)$ and $v \in V_j(G)$. For each vertex $v_z^{(i)} \in V_i(G)$, let $L(v_z^{(i)}, j)$ denote the set of all edge candidates $(v_z^{(i)}, v)$, where $v \in V_j(G)$. Finally, let $R(i, j)$ and $R'(i, j)$ denote the following two orders (which, indeed, are the crucial part of our construction):

$$\begin{aligned} R(i, j): & v_1^{(i)} \succ L(v_1^{(i)}, j) \succ \dots \succ v_{n'}^{(i)} \succ L(v_{n'}^{(i)}, j), \\ R'(i, j): & L(v_1^{(i)}, j) \succ v_1^{(i)} \succ \dots \succ L(v_{n'}^{(i)}, j) \succ v_{n'}^{(i)}. \end{aligned}$$

We construct a set V of $3h + 2(h + 1) \cdot \binom{h}{2}$ voters as follows.

1. For each color i , $(1 \leq i \leq h)$, construct one voter with orders $v_1^{(i)} \succ \dots \succ v_{n'}^{(i)} \succ d \succ \dots$.
2. For each pair of colors i, j , $(1 \leq i \neq j \leq h)$, construct $h - 1$ voters with orders $E(i, j) \succ d \succ \dots$, and another two voters, one with orders $R(i, j) \succ d \succ \dots$ and one with orders $R'(i, j) \succ d \succ \dots$.
3. Construct h voters with orders $d \succ \dots$ and h voters with orders $p \succ \dots$.

We claim that p can become a winner by adding at most $k := h + 2\binom{h}{2}$ candidates if and only if G has an order- h multi-colored clique (i.e., a clique containing a vertex for each color).

Simple calculation shows that if Q is a multi-colored clique of order h , then adding the vertex candidates and the edge candidates corresponding to Q makes p win.

Conversely, we observe that irrespective of how many candidates we add to the election, p cannot have more than h points. Thus, d and every added unregistered candidate *cannot* have more than h points in the final election. This implies that any size-at-most- $(h + 2\binom{h}{2})$ set A' of unregistered candidates that we add to the election must contain exactly one vertex candidate for each color and exactly one edge candidate for each (ordered) pair of colors. Further, if A' contains two vertex candidates u, v but not the edge candidate (u, v) , then, due to the orders $R(i, j) \succ d \succ \dots$ and $R'(i, j) \succ d \succ \dots$, either u or an edge candidate (u', v') (where $u' \in V_i(G)$, $v' \in V_j(G)$, but $(u', v') \neq (u, v)$) receives too many points, causing p not to win. To see why, note that $R(i, j)$ and $R'(i, j)$ contain all the candidates from $V_i(G)$ and $E(i, j)$. If we restrict those two preference orders to u and (u, v) , then they will become $u \succ (u, v)$ and the reverse one $(u, v) \succ u$. However, if we restrict them to u and (u', v') , then either they will both be $u \succ (u', v')$ or they will both be $(u', v') \succ u$. This completes the proof. \square

The Veto-CCAC case is quite intriguing. To see why, let us consider the following voting rule: Under *TrueVeto*, a candidate c is a winner if none of the voters ranks c last. It is quite easy to show that TrueVeto-CCAC is NP-complete, but it is also in FPT (when parameterized by the number of voters; an algorithm similar to that for Plurality-DCAC works). If a Veto election contained more candidates than voters, then at least one candidate would never be vetoed and, in effect, the election would be held according to the TrueVeto rule. This means that in the proof that Veto-CCAC is W[1]-hard, the election has fewer candidates than voters, even after adding the candidates (and keep in mind that the number of voters is the parameter!). Thus, the hardness of the problem lays in picking few spoiler candidates to add from a large group of them. If we were adding more candidates than voters, the problem would be in FPT.

In the combinatorial setting, there is a sharp difference between control by adding and by deleting candidates.

Theorem 2. *When parameterized by the number of voters, for Plurality and Veto, (1) COMB-DCAC is in FPT, (2) COMB-CCAC is W[1]-hard, and (3) COMB-CCDC and COMB-DCDC are para-NP-hard.*

Proof sketch for Plurality-COMB-DCDC. We reduce from SET COVER which, given a ground set $X = \{x_1, \dots, x_{n'}\}$, a family $\mathcal{S} = \{S_1, \dots, S_{m'}\}$ of subsets of X , and a non-negative integer h (taken to be the parameter), asks whether it is possible to pick at most h sets from \mathcal{S} so that their union is X . Given an instance I of SET COVER, we create an instance of Plurality-COMB-DCDC as follows. We let the candidate set be $C = \{p, d\} \cup X \cup \mathcal{S}$ (note that, depending on the context, we will use the symbol S_j , $1 \leq j \leq m'$, to denote both the set from \mathcal{S} and a set-candidate in the election). We introduce three voters with the following preference orders:

$$\begin{aligned} v_1: X \succ p \succ \dots, \\ v_2: d \succ \dots, \\ v_3: p \succ d \succ \dots. \end{aligned}$$

We set the bundling function κ so that for each set-candidate S_j , we have $\kappa(S_j) := \{S_j\} \cup \{x_i \mid x_i \in S_j\}$, and for every non-set candidate c , we have $\kappa(c) := \{c\}$.

We claim that the candidate d can be precluded from winning by deleting at most h bundles of candidates if and only if there are h sets from \mathcal{S} whose union is X .

Prior to deleting candidates, d , p , and one of the candidates from X are tied as winners. Deleting p would make d a unique winner, so the only way to defeat d is to ensure that v_1 gives its point to p . It is easy to see that we can assume that we only delete bundles of the set-candidates. To ensure that v_1 gives a point to p , all candidates from X must be deleted and, given our bundling function, this is possible (by deleting h bundles) if and only if the union of the sets corresponding to the deleted bundles is X . \square

For t -Approval and t -Veto with $t \geq 2$, there are fewer surprises and patterns are more clearly visible: In the non-combinatorial setting, constructive controls are $W[1]$ -hard and the destructive ones are in FPT. In the combinatorial setting, we have mostly hardness results.

Theorem 3. *When parameterized by the number of voters, for each fixed integer $t \geq 2$, for t -Approval and t -Veto, (1) (COMB)-CCAC, and CCDC are $W[1]$ -hard, (2) DCAC and DCDC are in FPT, (3) COMB-CCDC and COMB-DCDC are para-NP-hard, and (4) t -Approval-COMB-DCAC is $W[1]$ -hard.*

We conclude our discussion by claiming that in each of the $W[1]$ -hard cases discussed in this section we can, indeed, provide an XP algorithm. This means that these cases cannot be strengthened to para-NP-hardness results.

Theorem 4. *For each control type $\mathcal{K} \in \{CCAC, CCDC, \text{COMB-CCAC}, \text{COMB-DCAC}\}$, and for each fixed integer $t, t \geq 1$, each of t -Approval- \mathcal{K} and t -Veto- \mathcal{K} is in XP, when parameterized by the number of voters.*

5 Other Voting Rules

We focus on the voting rules Borda, Copeland $^\alpha$, and Maximin. The results are quite different from those for the case of t -Approval and t -Veto. Instead of FPT and $W[1]$ -hardness results, we obtain polynomial-time algorithms and para-NP-hardness results. Specifically, it has already been reported in the literature that there are polynomial-time algorithms for destructive candidate control in Borda Loreggia et al. [24], Copeland $^\alpha$ Faliszewski et al. [14], and Maximin (Faliszewski et al. [16]). For constructive candidate control, para-NP-hardness was already known for Copeland 0 and Copeland 1 Betzler and Uhlmann [2] and we establish it for the remaining values of α and for Borda and Maximin (in the latter case, only for CCAC; CCDC is known to be in P).

Theorem 5. *When parameterized by the number of voters, for Borda and Copeland $^\alpha$ ($0 \leq \alpha \leq 1$), CCAC and CCDC are para-NP-hard, and Maximin-CCAC is para-NP-hard.*

Proof sketch for Borda-CCDC. We reduce from the NP-complete problem CUBIC VERTEX COVER that given an undirected graph G , where each vertex has degree exactly three, and a non-negative integer h , asks whether there is a subset (*vertex cover*) of at most h vertices such that each edge is incident to at least one vertex in the subset.

Let $I = (G, h)$ be a CUBIC VERTEX COVER instance. We decompose $E(G)$ into four disjoint matchings (this is possible due to the computational variant of the classic graph-coloring result by Misra and Gries [27]) and rename the edges so that for each $\ell, 1 \leq \ell \leq 4$, the ℓ 'th of these matchings is $E^{(\ell)} = \{e_1^{(\ell)}, \dots, e_{m_\ell}^{(\ell)}\}$. We set $m' = m_1 + m_2 + m_3 + m_4 = \|E(G)\|$ and $n' = \|V(G)\|$. For each edge e , we arbitrarily order its vertices and we write $v'(e)$ and $v''(e)$ to refer to the first vertex and to the second vertex, respectively. For each $\ell, 1 \leq \ell \leq 4$, we write $EV^{(-\ell)}$ to mean the set of edges not in $E^{(\ell)}$ union the set of vertices not incident to

any of the edges in $E^{(\ell)}$. For each edge e , we define the following two orders over e , $v'(e)$, and $v''(e)$:

$$P(e): e \succ v'(e) \succ v''(e) \text{ and } P'(e): e \succ v''(e) \succ v'(e).$$

We form an election $E = (C, V)$, where $C = \{p, d\} \cup V(G) \cup E(G)$ and the voter set includes the following voters:

1. For each ℓ , $1 \leq \ell \leq 4$, we have the following two voters ($E^{(\ell)}$ is a matching so the orders are well-defined):

$$\begin{aligned} \mu(\ell): P(e_1^{(\ell)}) \succ \dots \succ P(e_{m_\ell}^{(\ell)}) \succ EV^{(-\ell)} \succ d \succ p, \text{ and} \\ \mu'(\ell): p \succ d \succ \overleftarrow{EV^{(-\ell)}} \succ P'(e_{m_\ell}^{(\ell)}) \succ \dots \succ P'(e_1^{(\ell)}). \end{aligned}$$

2. We have two voters, one with order $p \succ d \succ V(G) \succ E(G)$ and one with order $\overleftarrow{E(G)} \succ \overleftarrow{V(G)} \succ p \succ d$.

We claim that deleting at most h candidates can make p a winner if and only if there is a vertex cover of size h for G .

Initially, we have the following scores (to calculate them, note that—except for small asymmetries—our pairs of votes are reverses of each other): p has $5(n' + m') + 6$ points, d has $5(n' + m') + 4$ points, each edge candidate has $5(n' + m') + 7$ points, and each vertex candidate has $5(n' + m') + 2$ points. So, p has one point fewer than each edge candidate, but more points than the other ones.

Consider the effects of deleting candidates. Deleting d decreases the score of p by six, whereas it decreases the scores of each other candidate by five (so it is never beneficial to delete d). Further, if there is a solution that deletes some edge e , then a solution that is identical but instead of e deletes either $v'(e)$ or $v''(e)$ (it is irrelevant which one) is also correct. Now, let v be some vertex candidate. If we delete v , the score of each edge candidate e such that $v = v'(e)$ or $v = v''(e)$ decreases by six, and the score of each other remaining candidate decreases by five. Thus, there is a vertex cover of size h if and only if deleting vertices corresponding to the cover ensures p 's victory. \square

For combinatorial variants of candidate control, we only have one polynomial-time algorithm (for Maximin-COMB-DCAC); all the remaining cases are para-NP-hard. Our proofs mostly rely on the set-embedding technique. In particular, we prove that for every voting rule \mathcal{R} that satisfies the unanimity principle (that is, for each voting rule \mathcal{R} that chooses as the unique winner the candidate that is ranked first by all the voters), \mathcal{R} -COMB-CCDC is para-NP-hard.

Theorem 6. *Let \mathcal{R} be a voting rule that satisfies the unanimity principle. \mathcal{R} -COMB-CCDC is NP-hard even for the case of elections with just a single voter.*

Altogether, we have the following result.

Theorem 7. *When parameterized by the number of voters, for Borda, Copeland $^\alpha$ ($0 \leq \alpha \leq 1$), and Maximin, COMB- \mathcal{K} is para-NP-hard for each control type $\mathcal{K} \in \{\text{CCAC}, \text{CCDC}, \text{DCDC}\}$. For Borda and Copeland $^\alpha$ ($0 \leq \alpha \leq 1$), COMB-DCAC is para-NP-hard. On the contrary, Maximin-COMB-DCAC is polynomial-time solvable.*

In summary, for our more involved voting rules, constructive candidate control is hard even in the non-combinatorial setting, whereas destructive candidate control is tractable in the non-combinatorial setting, but becomes para-NP-hard in the combinatorial ones (with the exception of Maximin).

6 Outlook

Our work motivates several research directions. A particularly interesting one is to consider game-theoretic aspects of candidate control: Tractability results motivate studying more involved settings (e.g., consider a setting where two actors try to preclude two different candidates from winning; their goals might involve both cooperation and competition). Finally, taking a more general perspective, we believe that the case of few voters did not receive sufficient attention in the computational social choice literature and many other problems can (and should) be studied with respect to this parameter.

References

- [1] J. Bartholdi, III, C. Tovey, and M. Trick. How hard is it to control an election? *Mathematical and Computer Modeling*, 16(8/9):27–40, 1992.
- [2] N. Betzler and J. Uhlmann. Parameterized complexity of candidate control in elections and related digraph problems. *Theoretical Computer Science*, 410(52):43–53, 2009.
- [3] N. Betzler, J. Guo, and R. Niedermeier. Parameterized computational complexity of Dodgson and Young elections. *Information and Computation*, 208(2):165–177, 2010.
- [4] N. Betzler, R. Bredereck, J. Chen, and R. Niedermeier. Studies in computational aspects of voting—a parameterized complexity perspective. In *The Multivariate Algorithmic Revolution and Beyond*, volume 7370 of *LNCS*, pages 318–363. Springer-Verlag, 2012.
- [5] D. Binkle-Raible, H. Fernau, F. V. Fomin, D. Lokshtanov, S. Saurabh, and Y. Villanger. Kernel(s) for problems with no kernel: On out-trees with many leaves. *ACM Transactions on Algorithms*, 8(4):38, 2012.
- [6] C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.
- [7] F. Brandt, P. Harrenstein, K. Kardel, and H. G. Seedig. It only takes a few: on the hardness of voting with a constant number of agents. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS ’13)*, pages 375–382, 2013.
- [8] R. Bredereck, J. Chen, P. Faliszewski, A. Nichterlein, and R. Niedermeier. Prices matter for the parameterized complexity of shift bribery. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI ’14)*, pages 1398–1404, July 2014.
- [9] J. Chen, P. Faliszewski, R. Niedermeier, and N. Talmon. Combinatorial voter control in elections. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science*, pages 153–164, Aug. 2014.
- [10] B. Dorn and I. Schlotter. Multivariate complexity analysis of swap bribery. *Algorithmica*, 64(1):126–151, 2012.
- [11] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer-Verlag, 2013.

- [12] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of World Wide Web Conference (WWW-2001)*, pages 613–622, Mar. 2001.
- [13] G. Erdélyi, M. Fellows, J. Rothe, and L. Schend. Control complexity in Bucklin and Fallback voting. Technical Report arXiv:1103.2230 [cs.CC], Aug. 2012.
- [14] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Llull and Copeland voting computationally resist bribery and constructive control. *Journal of Artificial Intelligence Research*, 35:275–341, 2009.
- [15] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. Using complexity to protect elections. *Communications of the ACM*, 53(11):74–82, 2010.
- [16] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. Multimode control attacks on elections. *Journal of Artificial Intelligence Research*, 40:305–351, 2011.
- [17] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006.
- [18] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Anyone but him: The complexity of precluding an alternative. *Artificial Intelligence*, 171(5–6):255–285, 2007.
- [19] L. Hemaspaandra, R. Lavaee, and C. Menton. Schulze and ranked-pairs voting are fixed-parameter tractable to bribe, manipulate, and control. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '13)*, pages 1345–1346, May 2013.
- [20] R. Kannan. Minkowski’s convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, 1987.
- [21] H. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.
- [22] H. Liu and D. Zhu. Parameterized complexity of control problems in Maximin election. *Information Processing Letters*, 110(10):383–388, 2010.
- [23] H. Liu, H. Feng, D. Zhu, and J. Luan. Parameterized computational complexity of control problems in voting systems. *Theoretical Computer Science*, 410(27–29):2746–2753, 2009.
- [24] A. Loreggia, N. Narodytska, F. Rossi, K. Venable, and T. Walsh. Controlling elections by replacing candidates: Theoretical and experimental results. In *Proceedings of the 8th Multidisciplinary Workshop on Advances in Preference Handling*, pages 61–66, 2014.
- [25] N. Mattei, M. Pini, F. Rossi, and K. Venable. Bribery in voting over combinatorial domains is easy. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '12)*, pages 1407–1408, June 2012.
- [26] R. Meir, A. Procaccia, J. Rosenschein, and A. Zohar. The complexity of strategic behavior in multi-winner elections. *Journal of Artificial Intelligence Research*, 33:149–178, 2008.
- [27] J. Misra and D. Gries. A constructive proof of Vizing’s theorem. *Information Processing Letters*, 41(3):131–133, 1992.

- [28] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
- [29] D. Parkes and L. Xia. A complexity-of-strategic-behavior comparison between Schulze’s rule and ranked pairs. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI ’12)*, pages 1429–1435, July 2012.
- [30] J. Rothe and L. Schend. Challenges to complexity shields that are supposed to protect elections against manipulation and control: a survey. *Annals of Mathematics and Artificial Intelligence*, 68(1–3):161–193, 2013.
- [31] A. Schäfer, C. Komusiewicz, H. Moser, and R. Niedermeier. Parameterized computational complexity of finding small-diameter subgraphs. *Optimization Letters*, 6(5): 883–891, 2012.
- [32] L. Xia and V. Conitzer. Strategy-proof voting rules over multi-issue domains with restricted preferences. In *Proceedings of the 6th International Workshop On Internet And Network Economics (WINE ’10)*, pages 402–414, Dec. 2010.

Appendix

A Overview

In the appendix we provide all the proofs missing from the main text. However, the appendix is organized differently than the body of the paper. Instead of ordering the results with respect to the voting rules and problems studied, rather we sort them with respect to the proof technique. We provide next a connection between the organization of the body of the text and of the appendix. That is, for each of the theorems from the body of the text, we provide, in this appendix, a proof that points the reader to appropriate lemmas, proved in the following sections. We believe that this way, on one hand, the body of the paper tells how our results relate to each other, and the appendix, on the other hand, is concise and coherent.

A.1 Road Map

Below we provide a road map for the appendix. That is, for each of the theorems from the main body of the text, we list in which lemmas respective parts of the theorem are proved.

Theorem 1. *When parameterized by the number of voters, (1) for Plurality and Veto, DCAC and DCDC are both in FPT, (2) Plurality-CCAC and Veto-CCAC are both $W[1]$ -hard, and (3) Plurality-CCDC is in FPT, while Veto-CCDC is $W[1]$ -hard.*

Proof. The proof of this theorem is divided into the following lemmas:

- (1) The FPT results for DCAC and DCDC under Plurality and Veto are given in Lemma 27, Corollary 6, Lemma 31, and Corollary 4.
- (2) The $W[1]$ -hardness results for Plurality-CCAC and for Veto-CCAC are given in Lemmas 1 and 2.
- (3) The FPT result for Plurality-CCDC is given in Lemma 30 and the $W[1]$ -hardness result for Veto-CCDC is given in Lemma 3.

The $W[1]$ -hardness results use the multi-colored clique technique and the FPT results use the signatures technique and the brute-force technique. \square

Theorem 2. *When parameterized by the number of voters, for Plurality and Veto, (1) COMB-DCAC is in FPT, (2) COMB-CCAC is $W[1]$ -hard, and (3) COMB-CCDC and COMB-DCDC are para-NP-hard.*

Proof. The proof of this theorem is divided into the following lemmas:

- (1) The FPT results for COMB-DCAC under Plurality and Veto are shown in Corollary 5.
- (2) The $W[1]$ -hardness results for COMB-CCAC under Plurality and Veto follows from Lemmas 1 and 2.
- (3) The para-NP-hardness results for COMB-CCDC under Plurality and Veto follow from Corollary 3 and Lemma 15. The para-NP-hardness results for COMB-DCDC under Plurality and Veto are shown in Lemma 16 and Lemma 18.

The $W[1]$ -hardness results use the multi-colored clique technique, the para-NP-hardness results use the set-embedding technique, and the FPT results use the signatures technique. \square

Theorem 3. *When parameterized by the number of voters, for each fixed integer $t \geq 2$, for t -Approval and t -Veto, (1) (COMB)-CCAC, and CCDC are $W[1]$ -hard, (2) DCAC and DCDC are in FPT, (3) COMB-CCDC and COMB-DCDC are para-NP-hard, and (4) t -Approval-COMB-DCAC is $W[1]$ -hard.*

Proof. The proof of this theorem is divided into the following lemmas:

1. The $W[1]$ -hardness results for t -Approval-(COMB)-CCAC ($t \geq 2$) are shown in Corollary 1. The $W[1]$ -hardness results for t -Approval-CCDC ($t \geq 2$) are shown in Lemmas 5 and 6. The $W[1]$ -hardness results for t -Veto-(COMB)-CCAC ($t \geq 2$) are shown in Corollary 2. The $W[1]$ -hardness results for t -Veto-CCDC ($t \geq 2$) are shown in Lemma 4.
2. The FPT results for DCAC for t -Approval and t -Veto ($t \geq 2$) are shown in Lemma 28 and Corollary 4. The FPT results for DCDC for t -Approval and t -Veto ($t \geq 2$) are shown in Lemma 29.
3. The para-NP-hardness results for COMB-CCDC and COMB-DCDC are shown in Lemma 14, Lemma 17, Lemma 15, and Lemma 18.
4. The $W[1]$ -hardness result for t -Approval-COMB-DCAC is shown in Lemma 7.

The $W[1]$ -hardness results use the multi-colored clique technique, the para-NP-hardness results use the set-embedding technique, and the FPT results use the signatures technique. \square

Theorem 4. *For each control type $\mathcal{K} \in \{CCAC, CCDC, COMB-CCAC, COMB-DCAC\}$, and for each fixed integer $t, t \geq 1$, each of t -Approval- \mathcal{K} and t -Veto- \mathcal{K} is in XP, when parameterized by the number of voters.*

Proof. For the non-combinatorial part, the theorem follows from Lemma 32. For the combinatorial part, it follows from Lemma 33. \square

Theorem 5. *When parameterized by the number of voters, for Borda and Copeland $^\alpha$ ($0 \leq \alpha \leq 1$), CCAC and CCDC are para-NP-hard, and Maximin-CCAC is para-NP-hard.*

Proof. The proof of this theorem is divided into the following lemmas:

1. The para-NP-hardness result for Borda-CCAC is shown in Lemma 9.
2. The para-NP-hardness result for Borda-CCDC is shown in Lemma 8.
3. The para-NP-hardness result for Copeland $^\alpha$ -CCAC is shown in Lemma 11.
4. The para-NP-hardness result for Copeland $^\alpha$ -CCDC is shown in Lemma 12.
5. The para-NP-hardness result for Maximin-CCAC is shown in Lemma 10.

The results use the cubic vertex-cover technique. \square

Theorem 6. *Let \mathcal{R} be a voting rule that satisfies the unanimity principle. \mathcal{R} -COMB-CCDC is NP-hard even for the case of elections with just a single voter.*

Proof. This result is shown in Lemma 13. \square

Theorem 7. *When parameterized by the number of voters, for Borda and Copeland $^\alpha$ ($0 \leq \alpha \leq 1$), CCAC and CCDC are para-NP-hard, and Maximin-CCAC is para-NP-hard.*

Proof. The proof of this theorem is divided into the following lemmas:

1. The para-NP-hardness results for Borda are shown in Corollary 3, Lemma 19, and Lemma 20.
2. The para-NP-hardness results for Copeland $^\alpha$ (for $0 \leq \alpha \leq 1$) are shown in Lemma 22 and Lemma 21.
3. The para-NP-hardness results for Maximin are shown in Corollary 3, Lemma 24, and Lemma 23.

The results use the set-embedding technique. Finally, the polynomial-time algorithm for Maximin-COMB-DCAC is described in Theorem 8. \square

B Multi-Colored Clique Technique

In this section, we give the proofs based on the MULTI-COLORED CLIQUE technique. Specifically, we prove the following statements (all results are for the parameterization by the number of voters):

1. For each fixed integer $t \geq 1$ and for each voting rule $\mathcal{R} \in \{t\text{-Approval}, t\text{-Veto}\}$, \mathcal{R} -CCAC (and therefore also \mathcal{R} -COMB-CCAC) is $W[1]$ -hard.
2. For each fixed integer $t \geq 2$ and for each voting rule $\mathcal{R} \in \{\text{Veto}, t\text{-Approval}, t\text{-Veto}\}$, \mathcal{R} -CCDC is $W[1]$ -hard.
3. For each fixed integer $t \geq 2$ and for each voting rule $\mathcal{R} \in \{t\text{-Approval}, t\text{-Veto}\}$, \mathcal{R} -COMB-DCAC is $W[1]$ -hard.

All the proofs follow by reductions from MULTI-COLORED CLIQUE (hence the name of the technique) and are quite similar in spirit. Thus we start by providing some common notation and observations for all of them.

Let $I = (G, h)$ be a MULTI-COLORED CLIQUE instance with graph G and non-negative integer h . The vertices of G are partitioned into h sets, $V_1(G), \dots, V_h(G)$, each containing the vertices with a given color. Without loss of generality, we assume that each $V_i(G)$ contains the same number of vertices, denoted by n' , and we rename the vertices so that for each color i , $1 \leq i \leq h$, we have $V_i(G) = \{v_1^{(i)}, \dots, v_{n'}^{(i)}\}$. The task is to decide if there is a clique of order h where each vertex comes from a different set $V_i(G)$. Without loss of generality, we assume that each edge in G connects vertices with different colors and that the input graph contains at least two vertices.

In our reductions, given an instance $I = (G, h)$, we build elections with the following candidates related to the graph G (in addition to the candidates specific to a particular reduction). For each vertex $v \in V(G)$, we introduce a candidate denoted by the same symbol. For each edge $e = \{u, v\}$, we introduce two candidates (u, v) and (v, u) (while our original graph is undirected, for our construction we treat each undirected edge as two directed ones, one in each direction).

In the description of our preference orders, we will use the following orders over subsets of candidates. For each vertex $v_t^{(i)} \in V_i(G)$ of color i and each color $j, j \neq i$, we write $L(v_t^{(i)}, j)$ to denote the order obtained from

$$(v_t^{(i)}, v_1^{(j)}) \succ \dots \succ (v_t^{(i)}, v_{n'}^{(j)})$$

by removing those items (candidates) $(v_t^{(i)}, v_h^{(j)})$ for which there is no edge $\{v_t^{(i)}, v_h^{(j)}\}$ in the graph. Intuitively, $L(v_t^{(i)}, j)$ lists all the edge candidates for edges that include $v_t^{(i)}$ and go to vertices of color j (the particular order of these edges in $L(v_t^{(i)}, j)$ is irrelevant for our constructions).

Similarly, for each two colors $i, j, 1 \leq i, j \leq h, i \neq j$, we write $E(i, j)$ to mean the order

$$L(v_1^{(i)}, j) \succ L(v_2^{(i)}, j) \succ \dots \succ L(v_{n'}^{(i)}, j).$$

Intuitively, $E(i, j)$ lists all the edge candidates between the vertices from $V_i(G)$ and the vertices from $V_j(G)$ (note, however, that $E(i, j)$ and $E(j, i)$ are different).

The following two preference orders are crucial for the MULTI-COLORED CLIQUE technique. For each two colors, $i, j, 1 \leq i, j \leq h, i \neq j$, we define $R(i, j)$ and $R'(i, j)$ as follows:

$$\begin{aligned} R(i, j): v_1^{(i)} &\succ L(v_1^{(i)}, j) \succ \dots \succ v_{n'}^{(i)} \succ L(v_{n'}^{(i)}, j) \\ R'(i, j): L(v_1^{(i)}, j) &\succ v_1^{(i)} \succ \dots \succ L(v_{n'}^{(i)}, j) \succ v_{n'}^{(i)} \end{aligned}$$

The idea behind $R(i, j)$ and $R'(i, j)$ is as follows. Consider a setting where u is a vertex of color i and v is a vertex of color j (i.e., $u \in V_i(G)$ and $v \in V_j(G)$). Note that $R(i, j)$ and $R'(i, j)$ contain all the candidates from $V_i(G)$ and $E(i, j)$. If we restrict these two preference orders to candidates u and (u, v) , then they will become $u \succ (u, v)$ and $(u, v) \succ u$. That is, in this case they are reverses of each other. However, if we restrict them to u and some candidate (u', v') different than (u, v) , then either they will be both $u \succ (u', v')$ or they will be both $(u', v') \succ u$. Using this effect is at the heart of our constructions.

With the above setup, we are ready to prove the results of this section. We start with the adding candidates case and then, continue with the deleting candidates case.

Lemma 1. *Plurality-CCAC is $W[1]$ -hard, when parameterized by the number of voters.*

Proof. This is the first proof in which we employ the MULTI-COLORED CLIQUE technique. Let $I = (G, h)$ be our input instance of MULTI-COLORED CLIQUE with graph G and non-negative integer h . Let the notation be the same as described just prior to the lemma. We form an instance I' of Plurality-CCAC as follows. We let the registered candidate set C consist of two candidates, p and d , and we let the set A of unregistered candidates contain all the vertex candidates and all the edge candidates for G . We let p to be the preferred candidate. And we construct the election such that the current winner will be d . We introduce the following voters.

1. For each color $i, 1 \leq i \leq h$, we have one voter with preference order of the form

$$v_1^{(i)} \succ \dots \succ v_{n'}^{(i)} \succ d \succ \dots \succ p.$$

2. For each pair of colors i, j ($1 \leq i, j \leq h, i \neq j$), we have $h - 1$ voters with preference order of the form

$$E(i, j) \succ d \succ \dots \succ p.$$

3. For each pair of colors i, j ($1 \leq i, j \leq h, i \neq j$), we have two voters, one with preference order of the form

$$R(i, j) \succ d \succ \dots \succ p,$$

and one with preference order of the form

$$R'(i, j) \succ d \succ \dots \succ p.$$

4. We have h voters with preference order of the form

$$d \succ \dots \succ p,$$

and h voters with preference order of the form

$$p \succ \dots \succ d.$$

We set $k := h + 2\binom{h}{2}$. This complete the construction. Note that the total number of voters is $3h + 2(h + 1) \cdot \binom{h}{2}$ and the current winner is d having $(2h + 2(h + 1) \cdot \binom{h}{2})$ points.

We claim that it is possible to ensure that p becomes a winner by adding at most k candidates if and only if I is a “yes”-instance.

First, assume that I is a “yes”-instance of MULTI-COLORED CLIQUE and let Q be a size- h subset of vertices that forms a multi-colored clique in I . It is easy to see that if we add to our election the h candidates from Q and all the edge-candidates that correspond to edges between the candidates from Q , then, in the resulting election, each candidate (including p and d) will have h points (for example, each of the added vertex candidates will receive one point from the first group of voters and $h - 1$ points from the third group of voters). Thus everyone will win.

Now, assume that it is possible to ensure p 's victory by adding at most k candidates. Let A' be a subset of candidates such that $|A'| \leq h + 2\binom{h}{2}$ and adding the candidates from A' to the election ensures that p is a winner. Irrespective of the contents of the set A' , in the resulting election p will have h points. Thus, it follows that d must lose all points from the first three groups of voters implying that for each color i , $1 \leq i \leq h$, A' contains exactly one candidate from $V_i(G)$ and for each pair of colors i, j ($1 \leq i, j \leq h, i \neq j$), A' contains exactly one edge candidate (u, v) such that $u \in V_i(G)$ and $v \in V_j(G)$ (The fact that A' contains at least one candidate of each type follows because otherwise d would have more than h points; the fact that it contains exactly one of each type follows by a simple counting argument).

Now it suffices to prove that for each two vertex candidates $u, v \in A'$, we also have $(u, v) \in A'$. To show this, first observe there is a total of $h + 2(h + 1) \cdot \binom{h}{2} = h \cdot (h + 2\binom{h}{2}) = h \cdot k$ voters from the first three groups that will give points to the newly added candidates. Since each added candidate can have at most h points, it follows that $\|A'\| = k$ and each added candidate receives exactly h points. By the observations regarding preference orders $R(i, j)$ and $R'(i, j)$, $(u, v) \notin A'$, then, some vertex candidate or some edge candidate would be ranked first by at least two voters from the third group. If this were the case for an edge candidate, then—including the votes from the second group—this candidate would have more than h points and p would not be a winner. If this were the case for a vertex candidate (and neither of the edge candidates were ranked first by more than one of the voters in the third group), then this vertex candidate would receive at least h points from the voters in the third group and one point from the voters in the first group. Again, p would not be a winner. Thus, it must be that $(u, v) \in A'$. However, this proves that G has a multi-colored clique of order h . \square

Corollary 1. *For each fixed integer t , $t \geq 2$, t -Approval-CCAC is $W[1]$ -hard, when parameterized by the number of voters.*

Proof. It suffices to use the same proof as in the case of Lemma 1, but where for each voter, we introduce additional $t - 1$ registered dummy candidates which this voter ranks first (each voter ranks all the remaining dummy candidates last). In this way, each dummy candidate has exactly one point. The reasoning for the correctness proof works in the same way. \square

Lemma 2. *Veto-CCAC is $W[1]$ -hard, when parameterized by the number of voters.*

Proof. It suffices to use the same construction (and proof) as for the Plurality-CCAC case (Lemma 1), but with the following changes (notice that the order is important, that is, we perform the second modification only after we perform the first modification):

1. we swap the occurrences of p and d in every vote, and
2. we reverse each vote.

In effect, prior to adding candidates, p is vetoed by all but h voters and d is vetoed by exactly h voters. It is easy to verify that if we add vertex candidates and edge candidates that correspond to a multi-colored clique, then every candidate in the election is vetoed by exactly h voters and all the candidates are winners.

For the reverse direction, analogously as in the Plurality case, we note that we have to add exactly one vertex candidate of each color and exactly one edge candidate for each (ordered) pair of colors (otherwise p would receive more than h vetoes). To argue that for each two vertex candidates u and v that we add, we also have to add edge candidate (u, v) , we use the same reasoning as in the Plurality case, but pointing out that if some candidate receives two vetoes from the third group of voters, then some other one, altogether, receives fewer than h vetoes and p is not a winner. \square

Corollary 2. *For each fixed integer t , $t \geq 2$, t -Veto-CCAC is $W[1]$ -hard, when parameterized by the number of voters.*

Proof. It suffices to use the same proof as in Lemma 2, but we introduce $t - 1$ additional registered dummy candidates whose every voter ranks last. In this way, each dummy candidate receives exactly one veto from each voter, while p and d receive the same number of vetoes as in the election constructed in the proof for Lemma 2.

It is easy to verify that the arguments from that proof applies here as well. \square

Now, we move on to the deleting candidates case. We assume, without loss of generality, that the input graph is connected and contains at least two vertices

Lemma 3. *Veto-CCDC is $W[1]$ -hard, when parameterized by the number of voters.*

Proof. The proof follows by a reduction from the MULTI-COLORED CLIQUE problem. Let $I = (G, h)$ be our input instance with graph G and non-negative integer h and let the notation be as described in the introduction to the MULTI-COLORED CLIQUE technique section. We form an instance I' of Veto-CCDC as follows. We let the registered candidate set C consists all the vertex candidates and all the edge candidates for G , and the preferred candidate p . We construct the following groups of voters (set $H = 2 \binom{h}{2} = h \cdot (h - 1)$):

1. For each color i , $1 \leq i \leq h$, we introduce $2H - (h - 1)$ voters with preference order of the form

$$\cdots \succ p \succ v_1^{(i)} \succ \cdots \succ v_{n'}^{(i)}.$$

2. For each two colors i, j , $1 \leq i, j \leq h, i \neq j$, we introduce $2H - 1$ voters with preference order of the form

$$\cdots \succ p \succ E(i, j).$$

3. For each two colors, i, j , $1 \leq i, j \leq h, i \neq j$, we introduce two voters, one with preference order of the form

$$\cdots \succ p \succ R(i, j),$$

and one with preference order of the form

$$\cdots \succ p \succ R'(i, j).$$

4. We introduce $2H$ voters with preference order of the form $\cdots \succ p$.

We set the number k of candidates that can be deleted to $\|V(G)\| - h + 2\|E(G)\| - 2\binom{h}{2}$ (with the intention that one should delete all the candidates in the election except for the candidates corresponding to the vertices and edges of the multi-colored clique of order h). This completes the construction. Note that the total number of voters is

$$\begin{aligned} & (2H - (h - 1)) \cdot h + (2H - 1) \cdot H + 2H + 2H \\ &= 2H \cdot (H + h + 1) \\ &= 2(h - 1) \cdot h \cdot (h^2 + 1). \end{aligned}$$

Since the input graph is connected and contains at least two vertices, there is at least one candidate, either a vertex candidate or an edge candidate, which has fewer than $2H$ vetoes. Thus, p is currently not a winner.

We claim that p can become a winner by deleting at most k candidates if and only if I is a “yes”-instance.

First, it is easy to see that if G contains an order- h multi-colored clique and Q is the set of h vertices that form such a clique, then we can ensure that p is a winner. It suffices to delete all candidates from $V(G) \setminus Q$ and all the edge candidates except the ones of the form (u, v) , where both u and v belong to Q . In effect, each remaining candidate will have $2H$ vetoes and all the candidates will tie for victory. To see this, note that after deleting the candidates, p still receives $2H$ vetoes from the last group of voters. Now, for each color i , $1 \leq i \leq h$, consider the remaining vertex candidate of color i (call this vertex $v^{(i)}$). This candidate receives $2H - (h - 1)$ vetoes from the first group of voters. Further, there are exactly $h - 1$ voters in the third group that give one veto to $v^{(i)}$ each (these are the voters that correspond to the edges that connect $v^{(i)}$ with the other vertices of the clique). No other voter vetoes $v^{(i)}$. Now, for each two colors i and j , $1 \leq i, j \leq h, i \neq j$, consider the two edge candidates, call them (u, v) and (v, u) , whose corresponding edges are incident to vertices of color i (candidate u) and color j (candidate v). Both (u, v) and (v, u) still get $2H - 1$ vetoes from the second group of voters. It is also easy to see that each of them receives one veto from the third group of voters (for the case of (u, v) , this veto comes from the first voter corresponding to color choice (i, j) , and in the case of v , this veto comes from the first voter corresponding to color choice (j, i)).

Now, let us take care of the other direction. Assume that it is possible to ensure p 's victory by deleting at most k candidates. Prior to deleting any candidates, p has $2H$ vetoes and, of course, deleting candidates cannot decrease this number. Thus, we have to ensure that each non-deleted candidate has at least $2H$ vetoes.

Consider some two colors i and j ($1 \leq i, j \leq h, i \neq j$). Each edge candidate (u, v) (where the corresponding vertex u has color i and the corresponding vertex v has color j) appears below p in $2H - 1$ votes from the second group of voters and in 2 votes from the third one. If we keep two edge candidates, say (u', v') and (u'', v'') (where $u', u'' \in V_i(G)$ and $v', v'' \in V_j(G)$), then they are both ranked below p in the same $2H - 1$ votes from the second group and in the same two votes from the third one. If neither (u', v') nor (u'', v'') is deleted, then one of them will receive fewer than $2H$ vetoes. This means that for each two colors i and j , we have to delete all except possibly one edge candidate of the form (u, v) , where $u \in V_i(G)$ and $v \in V_j(G)$.

Similarly, for each color i , $1 \leq i \leq h$, each vertex-candidate from $V_i(G)$ appears below p in $2H - (h - 1)$ vetoes from the first group of voters and in $2(h - 1)$ votes from the third group. Each two candidates of the same color are ranked below p in the same votes in the first group. Thus, if two vertex-candidates of the same color were left in the election (after deleting candidates), then at least one of them would have fewer than $2H$ vetoes.

In consequence, and since we can delete at most $k = \|V(G)\| - h + 2\|E(G)\| - 2\binom{h}{2}$ candidates which means at least $h + H$ candidates except p must remain in the final election, if p is to become a winner, then after deleting the candidates the election must contain exactly one vertex candidate of each color, and exactly one edge-candidate for each ordered pair of colors.

Assume that p is among the winners after deleting candidates and consider two remaining vertex candidates u and v , $u \in V_i(G)$ and $v \in V_j(G)$ ($i \neq j$); they must exist by the previous observation. We claim that edge candidates (u, v) and (v, u) also must be remaining as well. Due to symmetry, it suffices to consider (u, v) . Careful inspection of voters in the third group shows that if (u, v) is not among the remaining candidates, then (using the observation regarding orders $R(i, j)$ and $R'(i, j)$) we have that the two voters from the third group that correspond to the color pair (i, j) either both rank u last or both rank the same edge candidate last. In either case, a simple counting argument shows that either u has fewer than $3H$ vetoes or the edge candidate corresponding to the ordered color pair (i, j) has fewer than $3H$ vetoes. In either case, p is not a winner. This shows that the remaining candidates correspond to an order- h multi-colored clique. \square

Lemma 4. *For each fixed integer $t \geq 1$, t -Veto-CCDC is $W[1]$ -hard, when parameterized by the number of voters.*

Proof. We use almost the same proof as in Lemma 3, but we add sufficiently many padding candidates to ensure that we can only delete vertex and edge candidates. Let $I = (G, h)$ be an input instance of MULTI-COLORED CLIQUE. Let $E' = (C', V')$ be the election created by the reduction from the proof of Lemma 3 on input I and set $k := \|V(G)\| - h + 2\|E(G)\| - 2\binom{h}{2}$.

We modify this election by extending C' to contain a set D of t dummy candidates, $D = \{d_1, \dots, d_t\}$, and modifying the voter collection V' as follows (recall that the number $\|V'\|$ of voters is a function polynomially bounded by h ; set $n' := \|V'\|$):

1. For each voter v in V' except the last group of voters, we modify v 's preference order to rank the dummies d_1, \dots, d_{t-1} last and d_t first.

2. For each voter v in the last group of V' , we rank all candidates from D such that v will have a preference order of the form

$$d_t \succ \dots \succ (D \setminus \{d_t\}) \succ p.$$

3. We add n' voters, all with preference order of the form

$$\dots \succ p \succ D.$$

It is easy to verify that each newly added candidate d_i , $1 \leq i \leq t-1$, has $2n'$ vetoes and d_t has n' vetoes. Since we assume the input graph to be connected and to have at least two vertices, at least one candidate from the edge and vertex candidates receives fewer vetoes than p . Thus, p is not a winner initially.

We claim that p (the preferred candidate from the proof of Lemma 3) can become a winner by deleting at most k candidates if and only if I is a “yes”-instance.

First, we note that if we delete any of the new dummy candidates from $D \setminus \{d_t\}$, then p certainly does not become a winner since p will have at least $n' + 2H$ vetoes and d_t will have exactly n' vetoes. If we delete dummy candidate d_t , then p will receive $2n'$ vetoes, but there is at least one remaining vertex or edge candidate which is not vetoed by the last group of voters and has hence, less than $2n'$ vetoes. In consequence, no dummy candidate can be deleted. Thus, neither of them will have fewer vetoes than p and (ignoring the dummy candidates) the election will behave as if it was held according to the Veto rule. The argument from the proof of correctness in Lemma 3 holds. \square

Lemma 5. *2-Approval-CCDC is $W[1]$ -hard, when parameterized by the number of voters.*

Proof. The proof is quite similar to that for the case of Veto-CCDC, but now the construction is a bit more involved. We give a reduction from the MULTI-COLORED CLIQUE problem. Let $I = (G, h)$ be our input instance with graph G and non-negative integer h , and let the notation be as described in the introduction to the MULTI-COLORED CLIQUE technique section. We form an instance I' of 2-Approval-CCDC based on I . We build our candidate set C as follows (we set $T = \|V(G)\| + \|E(G)\|$ with the intended meaning that T is an integer larger than the number of candidates that we can delete; we set $H := 2\binom{h}{2} = (h-1) \cdot h$):

1. We introduce the preferred candidate p .
2. We introduce T candidates b_1, \dots, b_T (these are the *blocker* candidates whose role, on one hand, is to ensure that p has to obtain a given number of points and, on the other hand, who prevent deleting too many candidates of other types).
3. For each vertex $v \in V(G)$, we introduce candidate v .
4. For each edge $\{u, v\} \in E(G)$, we introduce two candidates, (u, v) , and (v, u) .
5. We introduce two sets $D = \{d_1, \dots, d_h\}$ and $F = \{f_{(i,j)} \mid 1 \leq i \neq j \leq h\}$ of dummy candidates.

The set of voters consists of the following groups (we write B to refer to the preference order $b_1 \succ b_2 \succ \dots \succ b_T$):

1. We have $h + 3H$ voters, each with preference order of the form

$$B \succ \dots \succ p.$$

2. For each color i , $1 \leq i \leq h$, there are $3H + 1$ voters, where the first of them has preference order of the form

$$v_1^{(i)} \succ \dots \succ v_{n'}^{(i)} \succ p \succ B \succ \dots,$$

and the remaining ones have preference order of the form

$$v_1^{(i)} \succ \dots \succ v_{n'}^{(i)} \succ d_i \succ B \succ \dots.$$

3. For each pair i, j of distinct colors ($1 \leq i, j \leq h, i \neq j$), there are $3H + h - 1$ voters, where the first of them has preference order of the form

$$E(i, j) \succ p \succ B \succ \dots,$$

and the remaining ones have preference order of the form

$$E(i, j) \succ f_{(i, j)} \succ B \succ \dots.$$

4. For each pair i, j of distinct colors ($1 \leq i, j \leq h, i \neq j$), we introduce two voters with the following preference orders of the forms

$$p \succ R(i, j) \succ B \succ \dots$$

$$p \succ R'(i, j) \succ B \succ \dots$$

Note that the total number of constructed voters is polynomially bounded by h :

$$\begin{aligned} & h + 3H + (3H + 1) \cdot h + (3H + h - 1) \cdot H + 2H \\ = & 2h + 4H + 4H \cdot h + 3H^2. \end{aligned}$$

We set the number of candidates that can be deleted to $k := \|V(G)\| - h + 2\|E(G)\| - 2\binom{h}{2}$, with the intention that p can become a winner if and only if it is possible to delete all of the vertex candidates and all of the edge candidates except for the ones corresponding to a multi-colored clique of order h . We notice that if G indeed contains an order- h multi-colored clique Q , then deleting all the candidates in $V(G) \setminus Q$ and all the edge candidates of the form (u, v) where either $u \notin Q$ or $v \notin Q$ indeed ensures that p is a winner (in this case p , and all of the vertex and edge candidates have $h + 3H$ points each, and all of the blocker candidates have at most $h + 3H$ points each).

On the other hand, let us assume that it is possible to ensure p 's victory by deleting at most k candidates and let $C' \subseteq C$ be a set of at most k candidate such that p is a winner of $E' = (C \setminus C', V)$. Notice that $k < T - 1$ and so there are at least two blocker candidates that receive $h + 3H$ points each from the first group of voters. The only voters from whom p can obtain points after deleting at most k candidates are the ones in the second and third group and there are exactly $h + H$ of them (h in the second group and H in the third group). However, p can obtain the points from the second and the third groups of voters without, at the same time, increasing the score of the highest-scoring blocker candidate if and only if: (a) we delete all-but-one vertex-candidates of each color, and (b) for each pair i, j of distinct colors ($1 \leq i, j \leq h, i \neq j$) all-but-one edge-candidates of the form (u, v) , where $u \in V_i(G)$ and $v \in V_j(G)$. This means deleting exactly k candidates. Further, we claim that if p is a winner of E' , then for each two not-deleted vertex-candidates u and v , it must be the case that

both edge-candidates (u, v) and (v, u) are still in the election, meaning that there is an edge between u and v in the original graph. It suffices to consider the case of (u, v) (the case of (v, u) is symmetric). If instead of (u, v) the only not-deleted edge candidate for the pair of colors of u and v is some edge candidate (u', v') (where $(u', v') \neq (u, v)$), then one of the two following cases must happen: either u and v would receive more than $h - 1$ points from the fourth group, therefore would have more than $h + 2\binom{h}{2}$, causing p not be a winner, or (u', v') would receive more than 1 point from the fourth group, again causing p to not be a winner. Thus p can become a winner by deleting at most H candidates if and only if G contains a multi-colored clique of order h .

It is clear that the given reduction can be computed in polynomial time and the proof is complete. \square

Lemma 6. *For each fixed integer t , $t \geq 3$, t -Approval-CCDC is $W[1]$ -hard, when parameterized by the number of voters.*

Proof. Let $E' = (C', V')$ be the election constructed in the proof for Lemma 5. It suffices to use the same proof as for Lemma 5 except that now for each voter $v_i \in V'$ we introduce a group of $t - 2$ new dummy candidates, $d_1^i, d_2^i, \dots, d_{t-2}^i$, that are ranked first, and for each such introduced group, we introduce one yet two new dummies, c_1^i and c_2^i , and $\|V'\| - 1$ voters with preference order of the form (we write D_i to refer to the preference order $d_1^i \succ d_2^i \succ \dots \succ d_{t-2}^i$):

$$D_i \succ c_1^i \succ c_2^i \succ B \succ \dots.$$

These voters ensure that none of the new dummy candidates can be deleted without increasing the score of the highest-scoring blocker candidate. If a score of a highest-scoring blocker candidate increases, then the preferred candidate has no longer any chance of winning. If none of the new dummy candidate can be deleted, then the correctness proof works the same as the one given for Lemma 5.

The number of voters is still polynomially bounded by the clique order h . \square

We now move on to the combinatorial variant of destructive control by adding candidates, for t -Approval and t -Veto (for $t \geq 2$). In this case we still use a technique very similar to the proofs we have seen so far, but since we are in the combinatorial setting, the proofs can rely on the bundling function to ensure consistency between the added edge candidates and vertex candidates (indeed, for these cases, the analogous non-combinatorial problem is fixed-parameter tractable).

Lemma 7. *For each fixed integer $t \geq 2$, t -Approval-COMB-DCAC is $W[1]$ -hard, when parameterized by the number of voters.*

Proof. Given an instance of MULTI-COLORED CLIQUE, we construct an instance of t -Approval-COMB-DCAC. For this proof, it is more natural to create only one candidate for each edge, and not two “directed” ones. We let the set of registered candidates be of the form $C = \{p, d\} \cup D$, where D is the following sets of dummy candidates:

$$\begin{aligned} D = & \{d_z^{i,j} \mid i \neq j \in [h], z \in [t-1]\} \\ & \cup \{d_z^{(i)} \mid i \in [h], z \in [t-1]\} \\ & \cup \{e_z^{(i)} \mid i \in [h], z \in [t-1]\}. \end{aligned}$$

Candidate d is the despised one whose victory we want to preclude. We let the set of additional (unregistered) candidates be

$$A = V(G) \cup E(G).$$

That is, A contains all the vertex candidates and all the edge candidates. We set the bundling function κ so that for each edge candidate $e = (u, v)$, we have $\kappa(e) = \{e, u, v\}$, and for each vertex candidate v we have $\kappa(v) = \{v\}$. We introduce the following voters:

1. For each pair $\{i, j\} \subset [h]$, $i \neq j$, of distinct colors, we have one voter with the following preference order, we write $E(\{i, j\})$ to mean an arbitrarily chosen order over the edge candidates that link vertices of color i with those of color j):

$$E(\{i, j\}) \succ d_1^{\{i, j\}} \succ \dots \succ d_{t-1}^{\{i, j\}} \succ d \succ \dots.$$

Note that in the initial election, d gets a point from this voter, but it is sufficient (and we will make sure that it is also necessary) to add one candidate from $E(\{i, j\})$ to prevent d from getting this point.

2. For each color i , $1 \leq i \leq h$, we have a voter with the following preference order:

$$v_1^{(i)} \succ \dots \succ v_{n'}^{(i)} \succ d_1^{(i)} \succ \dots \succ d_{t-2}^{(i)} \succ p \succ d_{t-1}^{(i)} \succ \dots.$$

Note that in the initial election p gets a point from this voter, but if more than one candidate from $V_i(G)$ is added, then p does not gain this point.

3. For each number $i \in [h]$, we have a voter with the following preference order:

$$d \succ e_1^{(i)} \succ \dots \succ e_{t-1}^{(i)} \succ \dots.$$

Note that d gets one point from this voter.

First, prior to adding any candidates, d has $h + \binom{h}{2}$ points while p has h points and each of the dummy candidates has one point. We claim that it is possible to ensure that d is not a winner of this election by adding at most $k := \binom{h}{2}$ candidates if and only if G has a multi-colored clique of order h .

On one hand, easy calculation shows that if there is a multi-colored clique in G , then adding the edge-candidates corresponding to the edges of this clique ensures that d is not a winner.

For the other direction, let us assume that it is possible to ensure that d is not a winner by adding at most $\binom{h}{2}$ candidates. It is easy to see that p is the only candidate that can reach score higher than d this way. For this to happen, d must lose all the points that d initially got from the first group of voters, and p must still get all the points from the second group of voters. Moreover, adding voters corresponding to vertices does not help. Thus, this must correspond to adding $\binom{h}{2}$ edge candidates whose bundles do not add two vertices of the same color. That is, these $\binom{h}{2}$ added edge candidates must correspond to a multi-colored clique. \square

C Cubic Vertex Cover Technique

In this section we give the proofs based on the CUBIC VERTEX COVER technique. The idea is to prove para-NP-hardness via reductions from the CUBIC VERTEX COVER problem, using the fact that cubic graphs (that is, graphs where each vertex has degree three) can be easily encoded using a fixed number of votes. Formally, the CUBIC VERTEX COVER is defined as follows.

Definition 3. An instance of CUBIC VERTEX COVER consists of a graph $G = (V(G), E(G))$, where each vertex of G has degree exactly three, and a non-negative integer h . We ask if there is a subset (vertex cover) of at most h vertices such that each edge is incident to at least one vertex in the subset.

All our reductions in this section will use the following common setup. Let I be an instance of CUBIC VERTEX COVER with a graph G and non-negative integer h . By the classic result of Vizing, we know that there is an edge-coloring of G with four colors (that is, it is possible to assign one out of four colors to each edge so that no two edges incident to the same vertex have the same color). Further, it is possible to compute this coloring in polynomial time Misra and Gries [27]. This is equivalent to saying that it is possible to decompose the set of G 's edges into four disjoint matchings. Our reductions start by computing this decomposition and we rename the edges of G so that these four disjoint matchings are:

$$\begin{aligned} E^{(1)} &= \{e_1^{(1)}, \dots, e_{m_1}^{(1)}\} \\ E^{(2)} &= \{e_1^{(2)}, \dots, e_{m_2}^{(2)}\} \\ E^{(3)} &= \{e_1^{(3)}, \dots, e_{m_3}^{(3)}\} \\ E^{(4)} &= \{e_1^{(4)}, \dots, e_{m_4}^{(4)}\} \end{aligned}$$

We set $m' = m_1 + m_2 + m_3 + m_4 = \|E(G)\|$ and $n' = \|V(G)\|$. For each edge e of the graph, we arbitrarily order its vertices and we write $v'(e)$ and $v''(e)$ to refer to the first vertex and to the second vertex, respectively. For each ℓ , $1 \leq \ell \leq 4$, we write $E^{(-\ell)}$ to mean $E(G) \setminus E^{(\ell)}$. We write $V^{(-\ell)}$ to mean the set of vertices that are not incident to any of the edges in $E^{(\ell)}$.

The crucial point of our approach is to use the above decomposition to create eight votes (two for each matching) that encode the graph. We will now provide useful notation for describing these eight votes. For each edge e of the graph, we define the following four orders over e , $v'(e)$, and $v''(e)$:

$$\begin{aligned} P(e) &: e \succ v'(e) \succ v''(e), \\ P'(e) &: e \succ v''(e) \succ v'(e), \\ Q(e) &: v'(e) \succ v''(e) \succ e, \\ Q'(e) &: v''(e) \succ v'(e) \succ e. \end{aligned}$$

For each ℓ , $1 \leq \ell \leq 4$, we define the following orders over $V(G) \cup E(G)$:

$$\begin{aligned} A(\ell) &: P(e_1^{(\ell)}) \succ P(e_2^{(\ell)}) \succ \dots \succ P(e_{m_\ell}^{(\ell)}), \\ A'(\ell) &: P'(e_{m_\ell}^{(\ell)}) \succ \dots \succ P'(e_2^{(\ell)}) \succ \dots \succ P'(e_1^{(\ell)}), \\ B(\ell) &: Q(e_1^{(\ell)}) \succ Q(e_2^{(\ell)}) \succ \dots \succ Q(e_{m_\ell}^{(\ell)}), \\ B'(\ell) &: Q'(e_{m_\ell}^{(\ell)}) \succ \dots \succ Q'(e_2^{(\ell)}) \succ \dots \succ Q'(e_1^{(\ell)}). \end{aligned}$$

(Note that since each $E^{(\ell)}$ is a matching, each of the above orders is well-defined.) The first two of these families of orders (i.e., $A(\ell)$ and $A'(\ell)$) will be useful in the hardness proofs for the cases of deleting candidates and the latter two (i.e., $B(\ell)$ and $B'(\ell)$) in the hardness proofs for the cases of adding candidates. The intuitive idea behind orders $A(\ell)$ and $A'(\ell)$ ($B(\ell)$ and $B'(\ell)$) is that, at a high level, they are reverses of each other, but they treat edges and their endpoints in a slightly asymmetric way (we will describe this in detail in respective proofs).

Lemma 8. *Borda-CCDC is NP-hard, even for elections with only ten voters.*

Proof. We give a reduction from CUBIC VERTEX COVER (we use the notation as provided at the beginning of this section). Let I be our input instance that contains graph $G = (V(G), E(G))$ and non-negative integer h . We use the notation introduced in the beginning of the section. We form an election $E = (C, V)$, where $C = \{p, d\} \cup V(G) \cup E(G)$. We introduce the following ten voters:

1. For each ℓ , $1 \leq \ell \leq 4$, we have the following two voters:

$$\begin{aligned}\mu(\ell): A(\ell) &\succ E^{(-\ell)} \succ V^{(-\ell)} \succ d \succ p, \\ \mu'(\ell): p &\succ d \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ A'(\ell).\end{aligned}$$

2. We have one voter with preference order $p \succ d \succ V(G) \succ E(G)$ and one voter with preference order $\overleftarrow{E(G)} \succ \overleftarrow{V(G)} \succ p \succ d$.

We claim that p can become a winner of this election by deleting at most $k := h$ candidates if and only if there is a vertex cover of size h for G .

Let us first calculate the scores of all the candidates:

1. Candidate p has $5(n' + m') + 6$ points (that is, $4(n' + m' + 1)$ points from the first eight voters and $n' + m' + 2$ points from the last two voters).
2. Each vertex candidate v has $5(n' + m') + 2$ points (for each of the three pairs of voters $\mu(\ell)$, $\mu'(\ell)$, $1 \leq \ell \leq 4$, such that v is incident to some edge in $E^{(\ell)}$, v gets $n' + m'$ points; v gets $n' + m' + 1$ points from the remaining pair of voters in the first group and, additional, $n' + m' + 1$ points from the last two voters).
3. Each edge candidate e has $5(n' + m') + 7$ points (that is, $n' + m' + 3$ points from the pair of voters $\mu(\ell)$, $\mu'(\ell)$ such that $e \in E^{(\ell)}$, $n' + m' + 1$ points from each of the remaining voters in the first group, and $n' + m' + 1$ points from the last two voters).
4. Candidate d has $5(n' + m') + 4$ points (that is, $4(n' + m' + 1)$ points from the voters in the first group and $n' + m'$ points from the last two voters).

Clearly, prior to deleting any of the candidates, p is not a winner because edge candidates have higher scores. However, the score of p is higher than the score of the vertex candidates and the score of d .

We now describe how deleting candidates affect the scores of the candidates. Let v be some vertex candidate. Deleting v from our election causes the following effects: The score of each edge candidate e such that $v = v'(e)$ or $v = v''(e)$ decreases by six; the score of each other remaining candidate decreases by five. This means that if we delete h vertex candidates that correspond to a vertex cover of G , then the scores of p , d , and all the vertex candidates decrease by $5h$, while the scores of all the edge candidates decrease by at least $5h + 1$. As a result, we have p as a winner of the election.

On the other hand, assume that it is possible to ensure p 's victory by deleting at most h candidates. Deleting candidate d decreases the score of p by six, whereas it decreases the scores of each other candidate by five. Thus, we can assume that there is a solution that does not delete d . Similarly, it is easy to note that if there is a solution that deletes some edge e , then a solution that is identical but instead of e deletes either $v'(e)$ or $v''(e)$ (it is irrelevant which

one) is also correct. We conclude that it is possible to ensure p 's victory by deleting at most h vertex candidates. However, by the discussion of the effects of deleting vertex candidates and the fact that prior to any deleting each edge candidate has one point more than p , we have that these at-most- h deleted vertex candidates must correspond to a vertex cover of G . This completes the proof. \square

Lemma 9. *Borda-CCAC is NP-hard, even for elections with only ten voters.*

Proof. We give a reduction from CUBIC VERTEX COVER (we use the notation as provided at the beginning of this section). Given an instance (G, h) for CUBIC VERTEX COVER, we construct an instance for Borda-CCAC. We let the registered candidate set C be $\{p, d\} \cup E(G)$, and we let $V(G)$ be the set of unregistered candidates. We construct the following voters:

1. For each ℓ , $1 \leq \ell \leq 3$, we have the following two voters:

$$\begin{aligned}\mu(\ell): B(\ell) &\succ E^{(-\ell)} \succ V^{(-\ell)} \succ d \succ p, \\ \mu'(\ell): p &\succ d \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ B'(\ell).\end{aligned}$$

2. For $\ell = 4$, we have the following two voters:

$$\begin{aligned}\mu(\ell): B(\ell) &\succ E^{(-\ell)} \succ V^{(-\ell)} \succ d \succ p, \\ \mu'(\ell): d &\succ p \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ B'(\ell).\end{aligned}$$

3. We have two voters with preference orders

$$\begin{aligned}E(G) &\succ p \succ d \succ V(G) \\ p &\succ \overleftarrow{E(G)} \succ d \succ \overleftarrow{V(G)}.\end{aligned}$$

We claim that it is possible to ensure p 's victory by adding h candidates if and only if there is a vertex cover of size $k := h$ for G .

Note that at the beginning, p has $5m' + 5$ points, d has $4m' + 5$ points, and each edge candidate has $5m + 6$ points. Thus p is not a winner. Adding each unregistered vertex candidate v causes the scores of all the candidates to increase: For the edge candidates that include v as an endpoint this increase is by five points, whereas for all the other candidates this increase is by six points. Note that the last two voters always prefer the registered candidates to any vertex candidate. Thus, by simple counting, each of these h vertex candidates may obtain at most $4m' + 5h + 7$ points and will never obtain more points than p as long as $m' + h \geq 2$.

Thus, if we have a vertex cover of size h , then it is possible to ensure p 's victory by adding all the vertex candidates that correspond to this vertex cover. For the other direction, assume that it is possible to ensure p 's victory by adding at most h candidates and let S be such a set of candidates. For the sake of contradiction, assume that there is an edge candidate e which is not covered by some vertex candidate in S . It follows that the score of e is greater than the score of p , which is a contradiction. Thus S must correspond to a vertex cover in G . \square

Lemma 10. *Maximin-CCAC is NP-hard, even for elections with only ten voters.*

Proof. We give a reduction from CUBIC VERTEX COVER (we use the notation as provided at the beginning of this section). Given an instance (G, h) for CUBIC VERTEX COVER, we construct an instance for Maximin-CCAC. We let the registered candidate set C be $\{p\} \cup E(G)$, and we let $V(G)$ be the set of unregistered candidates. We construct the following voters:

1. For each $\ell, 1 \leq \ell \leq 4$, we have the following two voters:

$$\begin{aligned}\mu(\ell): B(\ell) &\succ E^{(-\ell)} \succ V^{(-\ell)} \succ p, \\ \mu'(\ell): p &\succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ B'(\ell).\end{aligned}$$

2. We have one voter with preference order $E(G) \succ p \succ V(G)$ and one voter with preference order $\overleftarrow{E(G)} \succ p \succ \overleftarrow{V(G)}$.

Let E be the thus-constructed election (including all the registered and unregistered candidates). We have the following values of the $N_E(\cdot, \cdot)$ function:

1. For each vertex $v \in V(G)$, we have $N_E(p, v) = 6$ (so $N_E(v, p) = 4$).
2. For each edge $e \in E(G)$, we have $N_E(p, e) = 4$ (so $N_E(e, p) = 6$).
3. For each vertex $v \in V(G)$ and each edge $e \in E(G)$ we have the following: If v is an endpoint of e , then $N_E(v, e) = 6$ (so $N_E(e, v) = 4$), and otherwise we have $N_E(v, e) = 5$ (so $N_E(e, v) = 5$).
4. For each two vertices, $v', v'' \in V(G)$, $N_E(v', v'') = 5$.
5. For each two edges, $e', e'' \in E(G)$, $N_E(e', e'') = 5$.

In effect, prior to adding the candidates, the score of p is four and the score of each edge candidate is five. Adding a vertex candidate v to the election does not change the score of p , but decreases the score of each edge candidate that has v as an endpoint to four. Further, this added vertex candidate has score four as well. Thus, it is easy to see that it is possible to ensure p 's victory by adding at most h candidates if and only if there is a size- h vertex cover for G . \square

Lemma 11. *For each rational number $\alpha, 0 \leq \alpha \leq 1$, Copeland $^\alpha$ -CCAC is NP-hard, even for elections with only twenty voters.*

Proof. We give a reduction from CUBIC VERTEX COVER (we use the notation as provided at the beginning of this section). Given an instance (G, h) for CUBIC VERTEX COVER, we construct an instance for Copeland $^\alpha$ -CCAC. We let the registered candidate set C be $\{p, d\} \cup E(G)$, and we let $V(G)$ be the set of unregistered candidates. We introduce the following voters:

1. For each $\ell, 1 \leq \ell \leq 4$, we construct four voters, two voters with the following preference order:

$$B(\ell) \succ E^{(-\ell)} \succ V^{(-\ell)} \succ d \succ p,$$

and two voters with the following preference order:

$$p \succ d \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ B'(\ell).$$

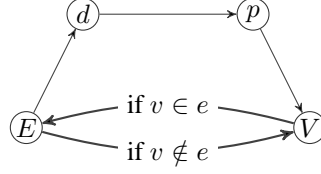


Figure 1: Illustration for the reduction used in the proof of Lemma 11. Each vertex in the graph correspond to a candidate or a set of candidates, and there is an arc going from a vertex u_1 to a vertex u_2 if u_1 beats u_2 in a head-to-head contest. Edges indicating ties are ignored. The main idea is that an edge candidate beats a vertex candidate if and only if the vertex candidate is one of the endpoint of the edge candidate.

2. One voter with the preference order $E \succ V \succ d \succ p$, and one voter with the preference order $d \succ p \succ \overleftarrow{E} \succ \overleftarrow{V}$.
3. One voter with the preference order $p \succ V \succ E \succ d$, and one voter with the preference order $\overleftarrow{E} \succ d \succ p \succ \overleftarrow{V}$.

We illustrate the results of head-to-head contests between the candidates in Figure 1. We claim that there is a vertex cover of size at most h for G if and only if p can become a winner of the election by adding at most $k := h$ candidates.

Consider a situation where we have added some subset A' of k candidates ($k \leq h$; take $k = 0$ to see the situation prior to adding any of the unregistered candidates). The candidates have the following scores:

1. p has score $\alpha m' + k$ (p ties head-to-head contests with all the edge candidates and wins all the head-to-head contests with the vertex candidates).
2. d has score $1 + \alpha k$ (d wins the head-to-head contest with p and ties all the head-to-head contests with the vertex candidates).
3. Each added vertex candidate v has score $3 + \alpha k$ (v ties the head-to-head contests with d and the remaining $k - 1$ vertex candidates and wins the head-to-head contests with the three edge candidates that are adjacent to v).
4. Each edge candidate e has score $\alpha m' + k + 1 - c(e)$, where $c(e)$ is the number of vertices from A' that are adjacent to e (e ties head-to-head contests with p and the remaining edge candidates and wins head-to-head contests with d and all the added vertex candidates except those that are adjacent to e).

In effect, it is easy to see that p is a winner of the election if and only if A' corresponds to a vertex cover of G . \square

Lemma 12. *For each rational number α , $0 \leq \alpha \leq 1$, Copeland $^\alpha$ -CCDC is NP-hard, even for elections with only twenty six voters.*

Proof. We give a reduction from CUBIC VERTEX COVER (we use the notation as provided at the beginning of this section). Given an instance for CUBIC VERTEX COVER (G, h) , we construct an instance for Copeland $^\alpha$ -CCDC. The candidate set contains the edge candidates,

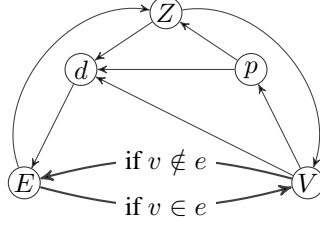


Figure 2: Illustration for the reduction used in the proof of Lemma 12. Each vertex in the graph correspond to a candidate or a set of candidates, and there is an arc going from a vertex u_1 to a vertex u_2 if u_1 beats u_2 in a head-to-head contest. Edges indicating ties are ignored. The main idea is that an edge candidate beats a vertex candidate if and only if the vertex candidate is one of the endpoint of the edge candidate.

the vertex candidate, the preferred candidate p , the dummy candidate d , and a set of further dummy candidates $Z = \{z_1, \dots, z_{m'+n'}\}$. We construct the following voters:

1. For each ℓ , $1 \leq \ell \leq 4$, we construct two voters with preference order:

$$A(\ell) \succ E^{(-\ell)} \succ V^{(-\ell)} \succ Z \succ d \succ p,$$

and two voters with preference order:

$$p \succ d \succ \overleftarrow{Z} \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ A'(\ell).$$

2. We also construct the following ten voters:

$$\begin{aligned} v_1 &: V \succ E \succ Z \succ d \succ p, \\ v'_1 &: p \succ d \succ \overleftarrow{Z} \succ \overleftarrow{V} \succ \overleftarrow{E}, \\ v_2 &: V \succ p \succ d \succ E \succ Z, \\ v'_2 &: \overleftarrow{E} \succ \overleftarrow{Z} \succ \overleftarrow{V} \succ p \succ d, \\ v_3 &: p \succ Z \succ d \succ V \succ E, \\ v'_3 &: \overleftarrow{E} \succ \overleftarrow{V} \succ p \succ \overleftarrow{Z} \succ d, \\ v_4 &: d \succ E \succ Z \succ V \succ p, \\ v'_4 &: p \succ \overleftarrow{V} \succ \overleftarrow{Z} \succ d \succ \overleftarrow{E}, \\ v_5 &: Z \succ V \succ E \succ d \succ p, \\ v'_5 &: p \succ d \succ \overleftarrow{E} \succ \overleftarrow{Z} \succ \overleftarrow{V}. \end{aligned}$$

Figure 2 illustrates the results of the head-to-head contests among the candidates. Prior to deleting any of the candidates, we have the following scores:

1. each edge candidate e has $m' + n' + \alpha m' + 2$ points (e wins head-to-head contests against all candidates in Z due to voters v_2 and v'_2 , wins head-to-head contests against its “incident” vertex candidates due to the first group of voters, and ties with p and the remaining edge candidates),

2. each vertex candidate u has $\alpha(n' - 1) + m' - 1$ points (u wins head-to-head contests against all edge candidates that are *not* “incident” to u due to voters from the first group, and ties with the remaining vertex candidates),
3. each candidate z from Z has $n' + 1 + \alpha(m' + n' - 1)$ points (z wins head-to-head contests against all vertex candidates and d due to voters v_3, v'_3, v_5, v'_5 , and ties with the remaining candidates from Z),
4. d has m' points (d wins head-to-head contests against all edge candidates due to voters v_4 and v'_4), and
5. p has $m' + n' + \alpha m' + 1$ points (p wins head-to-head contests against all candidates from Z due to voters v_3 and v'_3 , wins head-to-head contests against d due to voters v_2, v'_2, v_3, v'_3 , and ties with all edge candidates).

Thus, all edge candidates are co-winners, and p is not a winner because each edge candidate has one point more than it. However, p has more points than any other non-edge candidate. Note that in the input graph it holds that $m' = 3n'/2$

We claim that it is possible to ensure that p is a winner by deleting at most $k := h$ candidates if and only if there is a vertex cover of size h for G .

If there is a vertex cover for G of size h , then deleting the corresponding h vertices ensures that p is a winner. To see why this is the case, note that after deleting vertices corresponding to a vertex cover the score of p does not change, but the score of each edge candidate decreases by at least one. The scores of other candidates cannot increase, so p is a winner.

On the other hand, assume that it is possible to ensure that p is a winner by deleting at most h candidates. Deleting candidates cannot increase p 's score, so it must be the case that each edge candidate loses at least one point.

Observe that deleting candidates other than the vertex candidates will not make the edge candidates lose more than one point than p . The only possibility of deleting a candidate such that an edge candidate e loses a point but p does not is by deleting one of the vertex candidates, $v'(e)$ or $v''(e)$. Thus, if it is possible to ensure that p is a winner, we must delete vertices that correspond to a vertex cover. \square

D Set-Embedding Technique for Combinatorial Variants

In this section we give the proofs based on the Set-Embedding Technique for the combinatorial variants of our control problems. Specifically, we prove the following statements (all results are for the parameterization by the number of voters):

1. For each fixed integer $t \geq 1$ and for each voting rule $\mathcal{R} \in \{t\text{-Approval}, t\text{-Veto}, \text{Borda}, \text{Copeland}^\alpha \text{ (for } 0 \leq \alpha \leq 1), \text{Maximin}\}$, $\mathcal{R}\text{-COMB-DCDC}$ is para-NP-hard.
2. For each fixed integer $t \geq 1$ and for each voting rule $\mathcal{R} \in \{t\text{-Approval}, t\text{-Veto}, \text{Borda}, \text{Maximin}\}$, $\mathcal{R}\text{-COMB-CCDC}$ is para-NP-hard.
3. For each voting rule $\mathcal{R} \in \{\text{Borda}, \text{Maximin}\}$, $\mathcal{R}\text{-COMB-CCAC}$ is para-NP-hard.
4. For each voting rule $\mathcal{R} \in \{\text{Borda}, \text{Copeland}^\alpha \text{ (for } 0 \leq \alpha \leq 1)\}$, $\mathcal{R}\text{-COMB-DCAC}$ is para-NP-hard.

All proofs follow by reductions from SET COVER and use the bundling function to encode the sets from the SET COVER instances (hence the name of the technique). We start by providing some common notation and observations for all of them.

Let I be an input instance of SET COVER (which is NP-hard) with a ground set $X = \{x_1, \dots, x_{n'}\}$, a family $\mathcal{S} = \{S_1, \dots, S_{m'}\}$ of subsets of X , and a non-negative integer h . The task is to decide whether it is possible to pick at most h sets from \mathcal{S} so that their union is X . We assume that for each x_i there is some set S_j such that $x_i \in S_j$.

In our reductions we build elections with candidate sets that include the elements from X and the sets from \mathcal{S} . Specifically, for each element $x_i \in X$, we introduce a candidate with the same name, and for each set $S_j \in \mathcal{S}$, we introduce candidate s_j . We denote the set of all element candidates by X_{cand} and denote the set of all set candidates by $\mathcal{S}_{\text{cand}}$. Further, we will typically have candidates p and d . For the constructive cases, p will be the preferred candidate while for the destructive cases, d will be the despised one.

Unless we say otherwise, in each of our proofs we use a bundling function κ defined as follows: For each set candidate s_j , we have $\kappa(s_j) = \{s_j\} \cup \{x_i \mid x_i \in S_j\}$, and for each non-set-candidate c , we have $\kappa(c) = \{c\}$. We refer to this bundling function as *set-embedding bundling function*.

The general idea of our proofs is that to ensure p 's victory (for the constructive cases) or d 's defeat (for the destructive cases), one has to add/delete all the candidates from X_{cand} , and due to the bound on the number of candidates that we can add/delete, this has to be achieved by deleting the candidates from $\mathcal{S}_{\text{cand}}$ and relying on the bundling function.

With the above setup ready, we move on to proving our results.

D.1 Approval-Based Voting Rules

Constructive Control by Deleting Candidates. We first prove a general result which applies to all voting rules which satisfy the *unanimity* principle. A rule satisfies the *unanimity* principle if in each election where a unique candidate c is ranked first by all the voters, this candidate c is the unique winner.

Lemma 13. *Let \mathcal{R} be a voting rule that satisfies the unanimity principle. \mathcal{R} -COMB-CCDC is NP-hard even for the case of elections with just a single voter.*

Proof. Let the notation be as in the introduction to the set-embedding section. Given an instance $I := (X, \mathcal{S}, h)$ for SET COVER, we create an instance I' of \mathcal{R} -COMB-CCDC as follows. We construct an election $E = (C, V)$ where $C = \{p\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$ and where V contains a single voter with the following preference order:

$$X_{\text{cand}} \succ p \succ \mathcal{S}_{\text{cand}}.$$

We use the set-embedding bundling function. We claim that I is a “yes”-instance of SET COVER if and only if it is possible to ensure p 's victory by deleting at most h (bundles of) candidates.

On one hand, if I is a “yes”-instance of SET COVER, then I' is a “yes”-instance of \mathcal{R} -COMB-CCDC. Indeed, if \mathcal{S}' is a subfamily of \mathcal{S} such that $|\mathcal{S}'| \leq h$ and $\bigcup_{S_j \in \mathcal{S}'} S_j = X$, then it suffices to delete the candidates C' that correspond to the sets in \mathcal{S}' from the election to ensure that p is ranked first (and, by the unanimity of \mathcal{R} , is a winner).

On the other hand, assume that I' is a “yes”-instance of \mathcal{R} -COMB-CCDC. Since \mathcal{R} satisfies the unanimity property, the candidate ranked first by the only voter in our election is always

the unique winner. This means that if I' is a “yes”-instance of \mathcal{R} -COMB-CCDC, then there is a subset C' of candidates such that $p' \notin C'$ and $X \subseteq \bigcup_{c \in C'} \kappa(c)$. Without loss of generality, we can assume that C' contains only candidates from the set $\{s_1, \dots, s_m\}$ (if C' contained some candidate x_i , we could replace x_i with an arbitrary candidate s_j such that $x_i \in S_j$). However, this immediately implies that setting $S' := \{S_j \mid s_j \in C'\}$ results in a set cover of size at most h . Therefore I is a “yes”-instance of I . \square

As Plurality, Borda, Copeland ^{α} , and Maximin all satisfy the unanimity property, we conclude the following.

Corollary 3. *For each voting rule $\mathcal{R} \in \{\text{Plurality}, \text{Borda}, \text{Copeland}^\alpha, \text{Maximin}\}$, \mathcal{R} -COMB-CCDC is NP-hard even for the elections with only a single voter.*

We can slightly modify the reduction used in the proof of Lemma 13 to work for t -Approval (for $t \geq 2$).

Lemma 14. *For each fixed integer $t \geq 2$, t -Approval-COMB-CCDC is NP-hard even for elections with only a single voter.*

Proof. We build upon the proof of Lemma 13, but add $t - 1$ dummy candidates. Specifically, given an instance $I := (X, \mathcal{S}, h)$ for SET COVER, we create an instance I' of t -Approval-COMB-CCDC as follows. We construct an election $E = (C, V)$ where $C = \{p\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}} \cup D$, where $D = \{d_1, \dots, d_{t-1}\}$, and where V contains a single voter with the following preference order:

$$D \succ X_{\text{cand}} \succ p \succ \mathcal{S}_{\text{cand}}.$$

We use the bundling function as described in the introduction to the set-embedding section. We claim that I is a “yes”-instance of SET COVER if and only if it is possible to ensure p ’s victory by deleting at most h (bundles of) candidates.

To see the correctness of the argument, note that if there is a solution that ensures p by deleting a specific number of candidates, then there is also a solution that achieves the same and does not delete any of the dummy candidates (it is always at least as useful to delete one of the set candidates instead of a dummy one). \square

We can apply the same general reduction from Lemma 13 to t -Veto (for $t \geq 1$).

Lemma 15. *For each fixed integer $t \geq 1$, t -Veto-COMB-CCDC is NP-hard even for elections with only a single voter.*

Proof. Let the notation be as in the introduction to the set-embedding section. Given an instance $I := (X, \mathcal{S}, h)$ for SET COVER, we create an instance I' of t -Veto-COMB-CCDC as follows. We construct an election $E = (C, V)$ with candidate set:

$$C = \{p, z\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}} \cup D,$$

where $D = \{d_1, \dots, d_{t-1}\}$ is a set of dummy candidates (indeed, for $t = 1$, that is, for Veto, $D = \emptyset$), and with the voter collection V containing a single voter with the following preference order:

$$z \succ X_{\text{cand}} \succ \mathcal{S}_{\text{cand}} \succ D \succ p.$$

We use the set-embedding bundling function, with the added feature that $\kappa(z) = \mathcal{S}_{\text{cand}}$. We claim that I is a “yes”-instance of SET COVER if and only if it is possible to ensure p ’s victory by deleting at most $h + 1$ bundles.

Using similar reasoning as used in Lemma 14, it is easy to see that the only way of ensuring that p is a winner is to let all the remaining candidates receive no points at all. The only way to achieve this is to first delete up to h candidates from $\{s_1, \dots, s_m\}$ that correspond to a cover of the ground set and then to delete z . \square

Destructive Control by Deleting Candidates. We can also slightly modify the reductions from the previous section to work for the combinatorial destructive control case, although at the price of using more than one voter (indeed, this is unavoidable, because a candidate which is a t -Approval winner in an election with only one voter cannot be made a non-winner by deleting candidates). We first consider Plurality (we give a proof that uses three voters and this is, indeed, the smallest possible number of voters for which the proof works; if a candidate is a Plurality winner in a two-voter election, this candidate remains a winner irrespective which other candidates we delete).

Lemma 16. *Plurality-COMB-DCDC is NP-hard even for election with only three voters.*

Proof. Let the notation be as in the introduction to the set-embedding section. Given an instance $I := (X, \mathcal{S}, h)$ for SET COVER, we create an instance I' of Plurality-COMB-DCDC as follows. We construct an election $E = (C, V)$ where $C = \{p, d\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$, and where V contains three voters with the following preference orders:

$$\begin{aligned} X_{\text{cand}} &\succ p \succ \mathcal{S}_{\text{cand}} \succ d, \\ d &\succ X_{\text{cand}} \succ p \succ \mathcal{S}_{\text{cand}}, \text{ and} \\ p &\succ d \succ X_{\text{cand}} \succ \mathcal{S}_{\text{cand}}. \end{aligned}$$

We use the set-embedding bundling function. We claim that the despised candidate d can be precluded from winning by deleting at most h (bundles of) candidates if and only if there is a set cover of size h for I .

Prior to deleting any of the candidates, d , p , and one of the candidates from X are tied as winners. Since deleting candidates cannot make any candidate lose points and since deleting p will make d a unique winner, the only way of defeating d is by ensuring that the first voter gives its point to p . This means that all element candidates have to be removed from the election. By the same argument as in the previous proofs, doing so by deleting at most h candidates is possible if and only if I is a “yes”-instance of SET COVER. \square

We move on to consider t -Approval (for $t \geq 2$).

Lemma 17. *For each fixed integer $t \geq 2$, t -Approval-COMB-DCDC is NP-hard even for elections with only two voters.*

Proof. Let the notation be as in the introduction to the set-embedding section. Given an instance $I := (X, \mathcal{S}, h)$ for SET COVER, we create an instance I' of t -Approval-COMB-DCDC as follows. We construct an election $E = (C, V)$ with candidate set:

$$C = \{p, d\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}} \cup D \cup F,$$

where $D = \{d_1, \dots, d_{t-2}\}$ and $F = \{f_1, f_2, \dots, f_{t-1}\}$ are two sets of dummy candidates (note that D can be empty), and with the voter collection V containing two voters with the following preference orders:

$$\begin{aligned} d &\succ X_{\text{cand}} \succ D \succ p \succ \mathcal{S}_{\text{cand}} \succ F \text{ and} \\ p &\succ F \succ d \succ X_{\text{cand}} \succ \mathcal{S}_{\text{cand}} \succ D. \end{aligned}$$

We use the set-embedding bundling function. We claim that I is a “yes”-instance of SET COVER if and only if it is possible to preclude d from winning by deleting at most h (bundles of) candidates.

At the beginning, both d and p are winners (as well as some members of $X_{\text{cand}} \cup F$). Deleting p will make d gain one more point (from the second voter), making it impossible for d to lose. The same holds for the dummy candidates from set F . In other words, if we change the set of candidates that gain a point from the second voter, then d will obtain two points and will certainly be a winner. This implies that the only way of making d lose is to let either p or at least one candidate from F gain one point from the first voter. By construction of the first voter’s preference order, this is possible only for p , if and only if we delete all members of X_{cand} . As in the previous proofs, deleting them (through deleting at most h bundles of candidates) is possible if and only if I is a “yes”-instance of SET COVER. \square

We can also slightly modify the reduction from Lemma 14 to work for t -Veto.

Lemma 18. *For each fixed integer $t \geq 1$, t -Veto-COMB-DCDC is NP-hard even for elections with only a single voter.*

Proof. We use the same construction as used in Lemma 14 for t -Approval-COMB-CCDC but we reverse the preference order and swap p with d , the despised candidate:

$$S_{\text{cand}} \succ d \succ X_{\text{cand}} \succ D.$$

The crucial observation here is that with only one voter, the only way of preventing d from winning is to rank her within the last t positions. This means that all element candidates have to “disappear” from the election (one could also try deleting the dummy candidates, but it is never a mistake to “make disappear” the members of X_{cand} instead, through deleting the appropriate candidates in S_{cand}). Thus we can conclude that the set of deleted candidates contains the set candidates only. Clearly, if d is to be precluded from winning by deleting at most h candidates, this set must correspond to a set cover of size h . Since we assume that $h < \|S_{\text{cand}}\|$, there is at least one set element not deleted, and this will be a winner. \square

D.2 Borda Voting Rule

We now move on to considering Borda rule. Our proof approaches remain very similar to those used so far.

Lemma 19. *Borda-COMB-DCDC is NP-hard even for elections with only two voters.*

Proof. Let the notation be as in the introduction to the set-embedding section. Given an instance $I := (X, S, h)$ for SET COVER, we create an instance I' of Borda-COMB-DCDC as follows. We construct an election $E = (C, V)$ where $C = \{p, d, z\} \cup X_{\text{cand}} \cup S_{\text{cand}}$ and where V contains two voters with the following preference orders:

$$\begin{aligned} d &\succ X_{\text{cand}} \succ p \succ S_{\text{cand}} \succ z \text{ and} \\ p &\succ z \succ d \succ \overleftarrow{X_{\text{cand}}} \succ \overleftarrow{S_{\text{cand}}}. \end{aligned}$$

We use the set-embedding bundling function. We claim that I is a “yes”-instance of SET COVER if and only if it is possible to preclude d from winning by deleting at most h (bundles of) candidates.

For convenience, we calculate the scores of all the candidates:

1. d has $2\|\mathcal{S}_{\text{cand}}\| + 2\|X_{\text{cand}}\| + 2$ points.
2. p has $2\|\mathcal{S}_{\text{cand}}\| + \|X_{\text{cand}}\| + 3$ points.
3. each element candidate x_i has $2\|\mathcal{S}_{\text{cand}}\| + \|X_{\text{cand}}\| + 1$ points.
4. z has $\|\mathcal{S}_{\text{cand}}\| + \|X_{\text{cand}}\| + 1$ points.
5. each set candidate s_j has $\|\mathcal{S}_{\text{cand}}\|$ points.

Clearly, d has the highest number of points and, thus, is a winner.

Since both voters rank d ahead of the candidates in the set $X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$, no member of this set can have the score higher than d , irrespective which other candidates we delete. Similarly, irrespective which candidates we delete, z will never have score higher than d . We conclude that the only candidate that has a chance of defeating d , is p .

Since deleting candidates does not increase the scores of any of the remaining candidates, to ensure that d is not a winner, we have to guarantee that he or she loses at least $\|X_{\text{cand}}\|$ points (relative to p). This means that it is possible to ensure that d is not a winner if and only if it is possible to remove all the candidates from X_{cand} . However, this is possible if and only if I is a “yes”-instance of SET COVER. \square

Lemma 20. *Borda-COMB-CCAC and Borda-COMB-DCAC are both NP-hard even for elections with only two voters.*

Proof. Let the notation be as in the introduction to the set-embedding section. Given an instance $I := (X, \mathcal{S}, h)$ for SET COVER with $n := \|X_{\text{cand}}\|$, we create an instance I' of Borda-COMB-CCAC as follows. We construct the set of registered candidates $C = \{d, p\} \cup D$, where $D = \{d_1, \dots, d_n\}$. We construct the set of the unregistered candidates $A = X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$. We construct two voters with the following preference order:

$$d \succ D \succ p \succ \mathcal{S}_{\text{cand}} \succ X_{\text{cand}} \succ \dots \text{ and } \\ p \succ \overleftarrow{X_{\text{cand}}} \succ d \succ \mathcal{S}_{\text{cand}} \succ \overleftarrow{D} \succ \dots$$

We use the set-embedding bundling function. We claim that I is a “yes”-instance of SET COVER if and only if it is possible to preclude d from winning by adding at most h (bundles of) candidates.

note that d gets n points more than p from the first voter. Given a set cover of size h , we add the corresponding s_j ’s to the election. Simple calculation shows that in this case p and d tie as winners.

For the other direction, note that the relative scores of p and d in the first vote do not change irrespective which candidates we add. On the other hand, the relative scores of p and d to change in the second vote in the following way: For each unregistered candidate x_i added to the election, p ’s score increases by one but d ’s score remains unchanged. Thus, the only way to ensure that p is a winner is by bringing all the candidates from X_{cand} to the election. Doing so by adding at most h candidates is possible only if there is a size- h cover for I .

The construction for Borda-COMB-DCAC is the same, except that: First, we do not want p to win but d to lose (that is, we define d to be the despised candidate. Second, we define D to have only $n - 1$ dummy candidates. \square

D.3 Copeland $^\alpha$ and Maximin Voting Rules

We now move on to the cases of Copeland and Maximin voting rules. The flavor of our proofs changes a bit, albeit we still reduce from SET COVER.

We give the proofs for the case of Copeland $^\alpha$ rule even though, technically, they already follow from the non-combinatorial results. The reason is that this time we can give proofs that use much fewer voters.

Lemma 21. *Copeland $^\alpha$ -COMB-DCAC and Copeland $^\alpha$ -COMB-CCAC are NP-hard even for elections with only three voters.*

Proof. Let the notation be as in the introduction to the set-embedding section. Given an instance $I := (X, \mathcal{S}, h)$ for SET COVER with $n := \|X_{\text{cand}}\|$, we construct an instance for Copeland $^\alpha$ -COMB-DCAC. Since our reduction will produce an instance with an odd number of voters, the particular value of α is immaterial. We form the set of registered candidates:

$$C = \{d, p\} \cup D \cup F,$$

where d is the despised candidate (and we will want to ensure that p wins over d), and where $D := \{d_1, \dots, d_{n-2}\}$ and $F := \{f_1, \dots, f_{n-1}\}$ are two sets of dummy candidates. We let the set of of unregistered candidates be $A = X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$. Finally, we construct three voters with the following preference orders:

$$\begin{aligned} d &\succ D \succ p \succ F \succ X_{\text{cand}} \succ \mathcal{S}_{\text{cand}}, \\ p &\succ \overleftarrow{F} \succ \overleftarrow{X_{\text{cand}}} \succ \overleftarrow{D} \succ d \succ \overleftarrow{\mathcal{S}_{\text{cand}}}, \text{ and} \\ X_{\text{cand}} &\succ d \succ D \succ F \succ p \succ \mathcal{S}_{\text{cand}}. \end{aligned}$$

We use the set-embedding bundling function. We claim that I is a “yes”-instance of SET COVER if and only if it is possible to preclude d ’s victory by adding at most h (bundles of) candidates.

Prior to adding any of the candidates, we have the following scores:

1. d receives $2n - 2$ points (d wins head-to-head contests with all the remaining registered candidates).
2. p receives $n - 1$ points (p wins head-to-head contests with the members of F).
3. every dummy candidate $d_i \in D$ receives at most $2n - 3$ points (d_i wins head-to-head contests with all the members of F , with p , and—at most—all the remaining members of D).
4. every dummy candidate $f_i \in F$ receives at most $n - 2$ points (f_i wins head-to-head contests with—at most—the remaining members of F).

It is easy to verify through simple calculation that if there is a set cover for I of size at most h , then adding the members of $\mathcal{S}_{\text{cand}}$ that correspond to the cover ensures that d is not a winner (relative to d , p gets additional n points).

For the other direction, note that adding candidates to the election cannot decrease the score of any existing candidate. Thus, in order to beat d , we must add candidates to increase (relative to d) the score of some candidate. We make several observations:

1. The candidates in $\mathcal{S}_{\text{cand}}$ themselves do not contribute to the increase of a score of any candidate relative to p because all the other candidates (including d) win head-to-head contests against them.
2. The scores of the members of D do not change relative to the score of d irrespective which other candidates join the election.
3. By the first observation in this enumeration, the maximum possible increase of a score of candidate is by n points (if this candidate defeats all members of X_{cand} and members of X_{cand} join the election). Since all members of set F have score at most $n - 2$, neither of them can obtain score higher than d , irrespective which candidates we add.

As a final conclusion, we have that the only candidate that can possibly defeat d is p , and this happens only if all members of X_{cand} join the election. It is possible to ensure that this happens by adding at most h bundles of candidates if and only if there is a set cover for I of size at most h .

We use the same construction for the case of Copeland $^\alpha$ -CCAC, except that now p is the preferred candidate and we increase the size of D by one. \square

Lemma 22. *Copeland $^\alpha$ -COMB-DCDC is NP-hard even for elections with only three voters.*

Proof. The reduction is almost the same as the one given for Copeland $^\alpha$ -COMB-DCAC in Lemma 21, but even simpler. The candidate set is $C := \{p, d\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$.

We have three voters with the following preference orders (note that these are the same votes as in the proof of Lemma 21, restricted to the candidates present in our reduction, and with p and d swapped in each vote):

$$\begin{aligned}
p &\succ d \succ X_{\text{cand}} \succ \mathcal{S}_{\text{cand}}, \\
d &\succ X_{\text{cand}} \succ p \succ \mathcal{S}_{\text{cand}}, \text{ and} \\
\overleftarrow{X_{\text{cand}}} &\succ p \succ d \succ \overleftarrow{\mathcal{S}_{\text{cand}}}.
\end{aligned}$$

We use the set-embedding bundling function. We claim that I is a “yes”-instance of SET COVER if and only if it is possible to preclude d ’s victory by deleting at most h (bundles of) candidates.

The initial scores are:

1. d receives $\|\mathcal{S}_{\text{cand}}\| + \|X_{\text{cand}}\|$ points (d wins head-to-head contests against all the other candidates but p);
2. p receives $\|\mathcal{S}_{\text{cand}}\| + 1$ point (p wins head-to-head contests against d and all the members of $\mathcal{S}_{\text{cand}}$);
3. each member x_i of X_{cand} receives at most $\|\mathcal{S}_{\text{cand}}\| + \|X_{\text{cand}}\|$ (from head-to-head contests with p , all members of $\mathcal{S}_{\text{cand}}$, and the other members of X_{cand});
4. each member s_j of $\mathcal{S}_{\text{cand}}$ receives at most $\|\mathcal{S}_{\text{cand}}\| - 1$ points (from head-to-head contests with the other members of $\mathcal{S}_{\text{cand}}$).

Since deleting candidates cannot make any candidate gain more points, the only way of ensuring that d is not a winner is to make sure that d ’s score decreases relative to some other candidate. By the above list of scores, it is easy to see that the only candidate that may end

up with a score higher than d is p . This happens only if we remove all the members of X_{cand} . As in the previous proofs using the set-embedding technique, doing so by deleting at most h candidates is possible if and only if there is a set cover of size at most h for I . \square

Lemma 23. *Maximin-COMB-CCAC is NP-hard even for elections with only six voters.*

Proof. Let the notation be as in the introduction to the set-embedding section. Given an instance $I := (X, \mathcal{S}, h)$ for SET COVER with $n := \|X_{\text{cand}}\|$, we construct an instance for Maximin-COMB-CCAC. We let the set of registered candidates be $C := \{d, p\} \cup D$, where p is the preferred candidate and where $D := \{d_1, \dots, d_n\}$ is a set of dummy candidates. The unregistered candidate set is $A := X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$. We construct six voters with the following preference orders:

$$\begin{aligned} v_1: & p \succ x_1 \succ d_1 \succ \dots \succ x_n \succ d_n \succ \mathcal{S}_{\text{cand}}, \\ v_2: & p \succ x_n \succ d_n \succ \dots \succ x_1 \succ d_1 \succ \mathcal{S}_{\text{cand}}, \\ v_3: & x_1 \succ \dots \succ x_n \succ d_1 \succ \dots \succ d_n \succ p \succ \mathcal{S}_{\text{cand}}, \\ v_4: & d_n \succ \dots \succ d_1 \succ p \succ x_n \succ \dots \succ x_1 \succ \mathcal{S}_{\text{cand}}, \\ v_5: & x_1 \succ \dots \succ x_n \succ d_1 \succ \dots \succ d_n \succ p \succ \mathcal{S}_{\text{cand}}, \text{ and} \\ v_6: & d_n \succ \dots \succ d_1 \succ p \succ x_n \succ \dots \succ x_1 \succ \mathcal{S}_{\text{cand}}. \end{aligned}$$

(Note that the v_3 and v_5 have the same preference order and that v_4 and v_6 have the same preference order.) We use the set-embedding bundling function. We claim that I is a “yes”-instance of SET COVER if and only if it is possible to ensure p ’s victory by adding at most h (bundles of) candidates.

Prior to adding any of the candidates, p has two points and each candidate in D has three points. All the voters rank the members of $\mathcal{S}_{\text{cand}}$ last, so the presence of these candidates in the election does not change the scores of p and members of D . More so, member of $\mathcal{S}_{\text{cand}}$ themselves receive zero points each. On the other hand, if some candidate x_i appears in the election, then we have the following effects:

1. This candidate’s score is at most two (because only voters v_3 and v_5 prefer x_i to p).
2. The score of d_i becomes at most two (because only voters v_4 and v_6 prefer d_i to x_i).
3. The score of p does not change (because already v_1 and v_2 prefer p to x_i).

This means that if there is a set cover of size at most h for I , then adding the set candidates that correspond to this cover will bring all members of X_{cand} to the election and p will be among the winners.

For the other direction, note that on one hand, it is impossible to increase the score of p by adding candidates, and that for each d_i , the only way to decrease its score to at most two is to being x_i into the election.

For the other direction, notice that in order to let p , we must add candidates to the election to decrease the score of every element candidate x_i . and the only way to achieve this with adding at most k bundles is by adding the s_j corresponding to the set cover. This means that if it is possible to ensure p ’s victory by adding at most h candidates, it must be possible to add all members of X_{cand} into the election, and this means that there is a set cover of size at most h . \square

Lemma 24. *Maximin-COMB-DCDC is NP-hard, even for elections with only five voters.*

Proof. The proof is similar to the one given for Maximin-COMB-CCAC. Given an instance (X, \mathcal{S}, h) for SET COVER, we construct an instance for Maximin-COMB-DCDC. We construct a set of candidates $C := \{p, d, e\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$. We construct the following five voters:

$$\begin{aligned} \text{one voter: } & p \succ d \succ X_{\text{cand}} \succ e \succ \mathcal{S}_{\text{cand}}, \\ \text{two voters: } & d \succ X_{\text{cand}} \succ p \succ e \succ \mathcal{S}_{\text{cand}}, \\ \text{two voters: } & e \succ \overleftarrow{X_{\text{cand}}} \succ p \succ d \succ \overleftarrow{\mathcal{S}_{\text{cand}}}. \end{aligned}$$

We use the set-embedding bundling functions. We claim that I is a “yes”-instance of SET COVER if and only if it is possible to ensure that d is not a winner by deleting at most h (bundles of) candidates.

Let E be our election prior to deleting any of the candidates. The values of the $N_E(\cdot, \cdot)$ function are given in the table below (the entry for row a and column b gives the value of $N_E(a, b)$; we assume $i' \neq i''$ and $j' \neq j''$).

	p	d	e	$x_{i'}$	$s_{j'}$
p	-	3	3	1	5
d	2	-	3	3	5
e	2	2	-	2	5
$x_{i''}$	4	2	3	2 or 3	5
$s_{j''}$	0	0	0	0	2 or 3

We have the following scores of the candidates: p has one point (because of the members of X_{cand}), d has two points (because of p), e has two points (because of p, d , and the members of X_{cand}), the members of X_{cand} have two points each (because of d), and the members of $\mathcal{S}_{\text{cand}}$ have zero points each (because of all the other candidates).

It is easy to verify that if there is a set cover for I of size h , then deleting the set candidates corresponding to the cover deletes all the members of X_{cand} and ensures that p has three points, whereas d has only two. In effect, d certainly is not a winner.

Now consider the other direction. Since deleting a candidate can never decrease the score of any remaining candidate, the only way of making d lose is to increase some remaining candidate’s score.

Since for each candidate other than p , at least three voters prefer d to this candidate, only p has any chance of getting score higher than d . For this to happen, we need to ensure that all members of X_{cand} disappear. As in the previous set-embedding proofs, this is possible to do by deleting at most h candidates only if there is a set cover of size at most h for I . \square

E Signature Technique for Destructive Control

We now move on to our positive results obtained via the signatures technique. In this section we consider t -Approval and t -Veto elections only. We let d denote the despised candidate which we want to preclude from winning the election. Without loss of generality, we assume that d is a winner in the original election.

We observe that for given a candidate $p \neq d$, for a specific vote, an arbitrary candidate $c \notin \{p, d\}$ has only three possible relative positions compared to the candidates d and p (typically, the goal of candidate p will be to defeat the despised candidate): either c is in front of both, behind both, or in between them. Thus, given an election with n voters, each candidate can be characterized by a vector in $[3]^n$. We call such vectors *signatures*. Let $C' \subseteq C \cup A$ be a subset

of candidates (where C is the set of registered candidates and A is the set of unregistered ones; for the case of control by deleting candidates, we take $A = \emptyset$).

Definition 4 (C' -Signature). *Consider an election $(C \cup A, V)$ with $n := \|V\|$ and a set $C' \subseteq C \cup A$ of candidates. A size- n vector $\vec{\gamma} = (\gamma_1, \gamma_2, \dots, \gamma_n) \in [3]^n$ is a signature of candidate $c \in (C \cup A) \setminus C'$ if and only if for each voter $v_i \in V$, it holds that:*

$$\gamma_i = \begin{cases} 3 & \text{if for each } c' \in C', v_i: c \succ c', \\ 1 & \text{if for each } c' \in C', v_i: c' \succ c, \\ 2 & \text{otherwise.} \end{cases}$$

We will typically use $\{d, p\}$ -signatures. In this context, the above definition might be somewhat confusing, especially that value 2 of a signature vector can come both from voters with preference orders satisfying $p \succ c \succ d$ and from voters with preference orders satisfying $d \succ c \succ p$ (where c is the candidate in whose signature we are interested). However, note that for a given i 'th voter, if $\gamma_i = 2$, then this i 'th voter ranks p, d, c (where c is an arbitrary candidate with signature $\vec{\gamma}$) always in the same way. This will be a key observation in the proof of Lemma 25.

Using the signature technique, we will see that for the non-combinatorial destructive cases, all our control problems under approval-based election rules are fixed parameter tractable (when parameterized by the number of voters). We remark that the techniques used here also work for Plurality and Veto rules, but both rules are simple enough that brute-force approaches can be used to show their fixed-parameter tractability (Corollary 6). However, we do use the signature technique to obtain fixed-parameter tractability results for the combinatorial destructive control by adding candidates under both Plurality and Veto (Corollary 5).

E.1 Adding Candidates

To obtain fixed-parameter tractability results for the case where candidates are added (with the parameterization by the number n of voters), we use the following general scheme:

1. We guess one of the candidates and denote it by p . The role of this candidate is to defeat d , i.e., to obtain more points than d . Altogether there are $m := \|C\| + \|A\|$ candidates and we repeat our algorithm for each possible choice of p .
2. For each choice of p , we “kernelize” the input instance, that is, we bound the number of “relevant” candidates (or bundles) by a function of the parameter n , and search for an optimal solution in a brute-force manner over this “kernel”. This kind of kernelization is called *Turing kernelization* Binkele-Raible et al. [5], Schäfer et al. [31].

The idea of the kernelization is as follows. Say that we are dealing with destructive control by adding candidates under t -Approval (or t -Veto). For a given choice of p , adding exactly t (bundles of) candidates with the same $\{p, d\}$ -signature has the same effect on the relative scores of p and d as adding more than t such (bundles of) candidates. In effect, it suffices to keep at most t (bundles of) candidates with each signature. This results in having at most $t \cdot 3^n$ (bundles of) candidates.

Before we proceed with the formal presentation of the fixed-parameter algorithms, we introduce some notions and some basic observations.

Definition 5 (Relevant registered candidates). *Consider an instance of t -Approval-DCAC. We call a registered candidate relevant if this candidate receives at least one point. For the case of t -Veto-DCAC, we call a registered candidate relevant if this candidate receives at least one veto. We refer to those candidates that are not relevant as irrelevant.*

We observe that for the case of adding candidates (in contrast to the case of deleting candidates), under t -Approval, an irrelevant registered candidate can never beat the despised candidate d , irrespective of our actions. Thus we remove the irrelevant candidates.

As for the case of t -Veto, it suffices to focus on the case where d receives at least one veto and so do all the other registered candidates (in effect, all candidates are relevant). This is so for two reasons: First, if d were not vetoed by any voter, d would be a winner irrespective of our actions (we would have a trivial “no”-instance). Second, if d were vetoed by some voter but some registered candidate c was not vetoed by anyone, d already would not be a winner of the election (we would have a trivial “yes”-instance). All in all, we have the following observation.

Observation 1. *For each fixed t , $t \geq 1$, in nontrivial instances of t -Approval-DCAC and t -Veto-DCAC all the registered candidates are relevant.*

For each t -Approval-DCAC instance (t -Veto-DCAC instance) with n voters, at most $t \cdot n$ candidates are relevant. In the following sections we will show how to bound the number of unregistered candidates (separately for the non-combinatorial and combinatorial variants), using the notion of a signature. In effect, we will derive appropriate FPT algorithms.

Non-Combinatorial Variant. We note that if there is a way to preclude the despised candidate from being a winner in a given t -Approval or t -Veto election, it suffices to consider settings where we add at most t candidates with each given signature. This is formalized in the following lemma.

Lemma 25. *Consider an instance $I := ((C, V), A, d \in C, k)$ of t -Approval-DCAC (of t -Veto-DCAC), with the despised candidate d , and with some arbitrarily selected candidate $p \in C \cup A$. Let $\vec{\gamma}$ be some $\{d, p\}$ -signature for this election. Adding t unregistered candidates with signature $\vec{\gamma}$ has the same effect on the relative scores of p and d as adding more than t candidates with this signature.*

Proof. Let us focus on the case of t -Approval-DCAC. Let n be the number of voters in instance I . We have $\vec{\gamma} = (\gamma_1, \dots, \gamma_n)$. Consider the i 'th voter.

1. If $\gamma_i = 3$, then after adding t candidates with signature $\vec{\gamma}$, the i 'th voter will give 0 points to both p and d .
2. If $\gamma_i = 1$, then the i 'th voter will give the same number of points to p (resp. to d) as prior to adding candidates, irrespective how many candidates with signature $\vec{\gamma}$ we add.
3. If $\gamma_i = 2$, then either for each candidate c with signature $\vec{\gamma}$, the i 'th voter has preference order $p \succ c \succ d$, or for each candidate c with signature $\vec{\gamma}$, the i 'th voter has preference order $d \succ c \succ p$. In the first case, adding t (or more) candidates with signature $\vec{\gamma}$ will ensure that the i 'th voter gives zero points to d and gives the same number of points to p as prior adding the candidates. In the second case, the situation is the same, but with the roles of p and d swapped.

Summing over the points provided by all the voters, this proves that adding t candidates with a given signature $\vec{\gamma}$ has the same effect on the relative scores of p and d as adding any more such candidates. The argument for the case of t -Veto-DCAC is analogous. \square

Using this lemma, we can bound the number of unregistered candidates by a function depending only on n .

Lemma 26. *For each fixed integer $t \geq 1$, t -Approval-DCAC and t -Veto-DCAC admit Turing kernels of size $O(t \cdot 3^n)$.*

Proof. Consider an instance I of t -Approval-DCAC (of t -Veto-DCAC). Let n be the number of voters in the instance. As per our previous discussion, w.l.o.g., we can assume that the instances are nontrivial and that all the registered candidates are relevant. Thus, there are at most $t \cdot n$ registered candidates. By Lemma 25, for each choice of p it suffices to consider 3^n $\{d, p\}$ -signatures, and for each signature at most t candidates (the despised candidate d is given as part of the input). Altogether, for each choice of candidate p among the registered and unregistered candidates, we produce an instance of t -Approval-DCAC (of t -Veto-DCAC), with at most $t \cdot n$ registered candidates and at most $t \cdot 3^n$ unregistered ones (for each possible signature we keep up to t arbitrarily chosen unregistered candidates); in each instance we can add either the same number of candidates as in I , or one less, if p is an “added” candidate already. It is possible to preclude d from winning in the original instance if and only if it is possible to do so in one of the produced instances. \square

Using a brute-force approach on top of the kernelization given by Lemma 26, it is possible to solve both t -Approval-DCAC and t -Veto-DCAC in FPT time. Straightforward application of a brute-force search to each instance produced by Lemma 26 gives running time $O^*\left(\binom{3^n}{k}\right)$. However, it is easy to see that it never makes sense to add more than $t \cdot n$ candidates (intuitively, if we added more than $t \cdot n$ candidates, at least one would be irrelevant and we could as well not add him or her). Thus we can assume that $k \leq t \cdot n$. In effect, the straight-forward brute-force algorithm running on top of Lemma 26 has running time $O^*((3^n)^{t \cdot n})$. However, if we are willing to sacrifice more space, then we can obtain significantly better running times.

Lemma 27. *Plurality-DCAC can be solved in time $O(m \cdot n \cdot 2^n)$, using $O^*(2^n)$ space.*

Proof. Our algorithm uses a similar general structure as we used before. We assume that we are given a nontrivial instance, where all the registered candidates are relevant. First, we guess a candidate p whose goal is to defeat d and from now on we focus on a situation where we have both p and d , and the goal is to ensure that p gets more points than d . (If p is an unregistered candidate, we add p to the election, decrease the number of candidates that we can add by one, and proceed as if p was a registered candidate to begin with.)

We define a simplified notion of a candidate’s signature. A *signature* for an unregistered candidate a is a size- n binary vector $\vec{\tau} = (\tau_i)_i \in \{0, 1\}^n$, such that:

1. We have $\tau_i = 1$ if the i ’th voter ranks a ahead of all the registered candidates.
2. We have $\tau_i = 0$ if the i ’th voter ranks a below some registered candidate.

We define the signature of a set A' of unregistered candidates analogously: Value 1 at a given position means that some candidate from A' is ranked ahead of all the registered candidates and value 0 means that some registered candidate is ranked ahead all members of A' .

The crucial point of our algorithm is to compute a size- 2^n table $\vec{Z} := (\mathcal{Z}_{\vec{\tau}}) \in [k+1]^{2^n}$, such that for each signature $\vec{\tau} \in \{0, 1\}^n$, the $\mathcal{Z}_{\vec{\tau}}$ entry in the table is the size of the smallest subset $A_{\vec{\tau}}$ of unregistered candidates whose signature is $\vec{\tau}$.

With this new notion of signatures, we maintain a size- 2^n table $\vec{Z} := (\mathcal{Z}_{\vec{\tau}}) \in [k+1]^{2^n}$ which, for each signature $\vec{\tau} \in \{0, 1\}^n$, stores the minimum number $\mathcal{Z}_{\vec{\tau}}$ of unregistered candidates ($k+1$ indicates impossibility), such that there is a size- $\mathcal{Z}_{\vec{\tau}}$ subset $A^{(\vec{\tau})} \subseteq A \setminus \{p\}$ with signature $\vec{\tau}$. We compute this table as follows (our algorithm is slightly more complicated than necessary for the case of Plurality rule, but we will also use it as a base for more involved settings):

1. We initiate the table by setting $\mathcal{Z}_{\vec{\tau}} := 1$ if there is at least one unregistered candidate with signature $\vec{\tau}$, and we set $\mathcal{Z}_{\vec{\tau}} := k+1$ otherwise (value $k+1$ models the fact that it is impossible to achieve a given signature with at most k candidates).
2. For each unregistered candidate a we perform the following operations (for each two signatures $\vec{\tau}$ and $\vec{\tau}'$, we define a “merged” signature $\vec{\tau} \oplus \vec{\tau}'$ so that $\vec{\tau} \oplus \vec{\tau}' = (\max\{\tau_i, \tau'_i\})_{i \in [n]}$; in other words, we apply the coordinate-wise max operator):

- (a) We compute a ’s signature $\vec{\tau}_a$.
- (b) We compute a new table \mathcal{Z}' , by setting, for each signature $\vec{\tau}$:

$$\mathcal{Z}'_{\vec{\tau}} = \min(\{\mathcal{Z}_{\vec{\tau}}\} \cup \{\mathcal{Z}_{\vec{\tau}'} + 1 \mid \vec{\tau} = \vec{\tau}' \oplus \vec{\tau}_a\}).$$

- (c) We copy the contents of \mathcal{Z}' to \mathcal{Z} . (At this point, for each signature $\vec{\tau}$, $\mathcal{Z}_{\vec{\tau}}$ is the number of candidates in the smallest set composed of the so-far processed candidates that jointly have this signature.)
3. We pick a signature $\vec{\tau}$ such that $\mathcal{Z}_{\vec{\tau}}$ has a minimum value and adding the candidate set $A_{\vec{\tau}}$ that implements this signature ensures that p has more points than d (note that this last condition is easy to check: Given a signature $\vec{\tau}$, if the i ’th component τ_i is zero, then the i ’th voter gives one point to whoever this voter ranks first among the registered candidates; if τ_i is zero then the point goes to a candidate from $A_{\vec{\tau}}$, that is, neither to p or d). If $\mathcal{Z}_{\vec{\tau}}$ is smaller than the number of candidates that we can add, then we accept. Otherwise we reject (for this choice of p).

Let us first consider the algorithm’s running time. The most time-consuming part of the algorithm is the loop in the second step of the procedure computing the table \mathcal{Z} . For each out of at most m candidates, computing \mathcal{Z}' requires filling in $O(2^n)$ entries of the table. If we first copy the then-current contents of \mathcal{Z} to \mathcal{Z}' , and then perform the remaining updates, this can be done in time $O(m \cdot n \cdot 2^n)$. This dominates the running time of the remaining parts of the algorithm.

Now let us consider the correctness of the algorithm. Assume that we have guessed the correct candidate p and that there is subset of unregistered candidates $A' = \{a_1, \dots, a_\ell\}$ such that p has more points than d after we add candidate from A' , and ℓ is smaller or equal to the number of candidates that we can add. If $\vec{\tau}$ is the signature of the set A' , it is easy to verify that the algorithm indeed computes value $\mathcal{Z}_{\vec{\tau}} \leq \ell$. Further, if the algorithm accepts, then it is only by finding a verified solution. Thus the algorithm is correct. \square

We can apply the above ideas to the case of t -Approval and t -Veto as well.

Lemma 28. *For each fixed integer $t \geq 2$, t -Approval-DCAC can be solved in $\min\{O(t \cdot (3^n)^{t \cdot n}), O(m \cdot n \cdot t \cdot (t+1)^{t \cdot n})\}$ time.*

Proof. There are two means of solving our problem. We can either run the brute-force algorithm on top of Lemma 26, obtaining running time $O(t \cdot 3^{n \cdot t \cdot n})$, or we can use a variant of the algorithm from Lemma 27. Below we describe how to adapt the algorithm from Lemma 27.

We use the same algorithm as in Lemma 27, but we use a somewhat more involved notion of a signature and of the merging operator \oplus . If we have n voters, then an unbounded signature of a set A' of unregistered candidates is an n -dimensional vector $\vec{\tau}$, whose i 'th entrance is a t -dimensional vector τ_i defined as follows: The j 'th entry of τ_i contains the number of candidates in A' that the i 'th voter prefers to all but $j - 1$ registered candidates. Now a signature of a set A' is its unbounded signature where all entries greater than t are replaced by t . Altogether, there are $(t+1)^{t \cdot n}$ signatures.

Given two signatures, $\vec{\tau}'$ and $\vec{\tau}''$, we define their merge, $\vec{\tau} = \vec{\tau}' \oplus \vec{\tau}''$, as follows: For each i , $1 \leq i \leq n$, vector τ_i is computed by first calculating the component-wise sum of vectors τ'_i and τ''_i , and then replacing with t each entry greater than t . It is easy to see that if A' and A'' are two disjoint sets of candidates with signatures $\vec{\tau}_{A'}$ and $\vec{\tau}_{A''}$, then $\vec{\tau}_{A'} \oplus \vec{\tau}_{A''}$ is a signature of their union. (Note that In our algorithm we apply operator \oplus to “signatures of disjoint sets of candidates” only.)

It is straightforward to verify that given a signature of a subset A' of unregistered candidates, we can compute the scores of candidates p and d . This suffices to describe our algorithm and to justify its correctness. The running time is $O(m \cdot n \cdot t \cdot (t+1)^{t \cdot n})$ (it is calculated in the same way as in the proof of Lemma 27, except now we have more signatures and the components of the signatures are t -dimensional vectors). \square

Adapting the algorithms in a straight-forward way (basically by inverting, or reversing, the signatures) used for Lemma 27 and Lemma 28, we can show a similar statement for veto-based voting rules.

Corollary 4. *For each fixed integer $t \geq 1$, t -Veto-DCAC can be solved in $\min\{O(t \cdot (3^n)^{t \cdot n}), O(m \cdot n \cdot t \cdot (t+1)^{t \cdot n})\}$ time.*

Combinatorial Variant. The situation changes a little bit when instead of adding candidates we are adding bundles of candidates. We cannot bound the number of bundles for general approval-based (or veto-based) voting rules (where $t \geq 2$) by using the signature techniques since bundles with the same signature but with different sizes may have different effects on the score difference between the despised candidate d and a specific guessed candidate p (indeed, the corresponding combinatorial variants are computationally intractable, as shown in Theorem 3). However, for Plurality and for Veto, only the first (or the last) position gets a point. This allows us to use our non-combinatorial algorithms.

Corollary 5. *Plurality-COMB-DCAC and Veto-COMB-DCAC are fixed-parameter tractable.*

Proof. For the case of Plurality, it suffices to use, for example, the same algorithm as in Lemma 27, but with the following changes:

1. For each choice of candidate p , we also consider each way of adding p to the election, if p was unregistered (p might belong to several different bundles and we try each possibility).

2. Each unregistered candidate's signature is replaced by the signature of the set of candidates in its bundle.

Since under Plurality each voter gives a point only to whoever this voter ranks first, this strategy suffices. The case of Veto rule is handled analogously. \square

E.2 Deleting Candidates

The (Turing) kernelization approach for the case of adding candidates cannot be easily transferred to the case of deleting candidates. This is because we cannot upper-bound the number of candidates that have to be deleted in terms of the number n of the voters. However, applying our signature technique followed by casting the remaining task as an integer linear program (ILP), we can show fixed-parameter tractability (for our parameterization by the number of voters).

We now describe our approach. Let us fix a positive integer t and let $((C, V), d, k)$ be an instance of t -Approval-DCDC, where $V = (v_1, v_2, \dots, v_n)$. (We focus on the case of t -Approval and later we will argue how to adapt the results to apply to the case of t -Veto.) We guess a candidate p , whose role is to defeat the despised candidate d . For each such candidate p we do the following. First, we make an initial brute-force search: For each voter, we “guess” one of at most four possible choices of how d and p would gain points after our action of deleting candidates:

1. choice one: only d receives one point,
2. choice two: only p receives one point,
3. choice three: both candidates receive one point, and
4. choice four: neither p or d receive a point.

We record our guesses in vector $\vec{\delta}$. For each guessed p and $\vec{\delta}$, we check if giving the points according to our guesses in $\vec{\delta}$ guarantees that p has more points than d . If so, we run an integer linear program to verify if it is at all possible to ensure that every voter gives points to candidate p and d as described by vector $\vec{\delta}$, and to compute the smallest number of candidates we have to delete to ensure this. The complete procedure, for the case of t -Approval-DCDC, is given as Algorithm 1.

Lemma 29. *For each fixed integer $t \geq 1$, t -Approval-DCDC and t -Veto are both fixed-parameter tractable when parameterized by the number of voters.*

Proof. We start by considering the case of t -Approval-DCDC. The running time for Algorithm 1 is easy to verify: we guess a candidate p and a possible way of giving p and d points, followed by running an ILP. Therefore, the running time is $O(m \cdot 4^n)$ times the cost of running the ILP. The ILP has 3^n variables and $(3^n + 2n)$ constraints. Thus, employing the famous result by Lenstra, Jr. [21], our algorithm runs in $O^*(4^n \cdot f(n))$ where f is a function that describes the running time of the ILP solver and solely depends on n Lenstra, Jr. [21], Kannan [20].

To prove the correctness of the algorithm, it suffices to show the correctness of the ILP program for a given guess of p and $\vec{\delta}$. First, the constraint in Line 25 ensures that we do not delete more candidates with a given $\{d, p\}$ -signature than there are present in the election. The remaining signature verify that we can implement vector $\vec{\delta}$. For each i , $1 \leq i \leq n$, we verify if it is possible to implement guess δ_i :

Algorithm 1: FPT algorithm for t -Approval-DCDC.

Input:
 $((C, V), d, k)$ — input: an instance of t -Approval-DCDC
 p — a guessed candidate who is to defeat d

```

1 foreach  $\vec{\delta} = (\delta_1, \delta_2, \dots, \delta_n) \in [4]^n$  with  $|\{i \mid \delta_i = 1\}| < |\{i \mid \delta_i = 2\}|$  do
2   — Run ILP for each sane  $\vec{\delta}$  such that  $p$  beats  $d$ .
3   foreach  $i \in [n]$  do
4     if SanityCheck( $\delta_i$ ) = false then
5       └ Next  $\vec{\delta}$ ;
6   if  $p$  has more points than  $d$  when  $p$  and  $d$  receive points as described by  $\vec{\delta}$  and there is a solution for
7     ILP( $\vec{\delta}$ ) then
8       └ accept;
9 reject;

9 SanityCheck( $\delta_i$ )
10 if  $\delta_i = 1$  and ( $v_i : p \succ d$ ) then
11   —  $\delta_i = 1$ : only  $d$  gains one point.
12   └ return false;
13 if  $\delta_i = 2$  and ( $v_i : d \succ p$ ) then
14   —  $\delta_i = 2$ : only  $p$  gains one point.
15   └ return false;
16 return true;

17 ILP( $\vec{\delta} = (\delta_1, \delta_2, \dots, \delta_n)$ ):
18   Variables
19    $\forall \vec{\gamma} \in [3]^n : x_{\vec{\gamma}}$  — # deleted candidates with  $\{d, p\}$ -signature  $\vec{\gamma}$ 
20   Constants
21    $\forall \vec{\gamma} \in [3]^n : z_{\vec{\gamma}}$  — # existing candidates with  $\{d, p\}$ -signature  $\vec{\gamma}$ 
22   Objective
23    $\sum_{\vec{\gamma}} x_{\vec{\gamma}} \leq k$ 
24   Constraints
25    $\forall \vec{\gamma} \in [3]^n : x_{\vec{\gamma}} \leq z_{\vec{\gamma}}$ 
26    $\forall i \in [n] :$ 
27     if  $\delta_i = 1$  or  $\delta_i = 2$  then
28       —  $v_i : d \succ p$  and only  $d$  gains one point, or
29       —  $v_i : p \succ d$  and only  $p$  gains one point
30        $\sum_{\forall \vec{\gamma} : \gamma_i = 3} (z_{\vec{\gamma}} - x_{\vec{\gamma}}) \leq t - 1$ 
31        $\sum_{\forall \vec{\gamma} : \gamma_i = 3 \vee \gamma_i = 2} (z_{\vec{\gamma}} - x_{\vec{\gamma}}) \geq t - 1$ 
32     else if  $\delta_i = 3$  then
33       — Both  $d$  and  $p$  gain one point each
34        $\sum_{\forall \vec{\gamma} : \gamma_i = 3 \vee \gamma_i = 2} (z_{\vec{\gamma}} - x_{\vec{\gamma}}) + 2 \leq t$ 
35     else
36       — No one gains one point
37      $\sum_{\forall \vec{\gamma} : \gamma_i = 3} (z_{\vec{\gamma}} - x_{\vec{\gamma}}) \geq t$ 

```

1. If $\delta_i = 1$ (i.e., d gains a point from the i 'th voter but p does not) then according to our sanity check (SanityCheck) we have that v_i prefers d over p . Thus, after the candidate deletion, d must be ranked in the first t positions (Line 30) and p must be ranked behind the t 'th position (Line 31).

Algorithm 2: A generic brute-force search algorithm for the FPT results.

Input:
 (C, V) — an election

```

1 BruteForceSearch( $a$ ):
2   foreach  $\vec{b} \in [2]^n$  do
3     Delete all candidates in CollectCands( $\vec{b}, a$ );
4     Do appropriate task;

5 CollectCands( $\vec{b} := (b_1, b_2, \dots, b_n), a$ ):
6    $C' \leftarrow \emptyset$ ;
7   foreach  $i \in [n]$  do
8     if  $b_i = 1$  then
9        $C' \leftarrow C' \cup \{c \in C \setminus C' \mid v_i : c \succ a\}$ ;
10  return  $C'$ ;
```

2. If $\delta_i = 2$ which means that only p gains one point, then v_i prefers p over d . Thus, after the candidate deletion, p must be ranked in the first t positions (Line 30) and d must be ranked behind the t 'th position (Line 31).
3. If $\delta_i = 3$, then both candidates gain one point each and must be ranked in the first t positions (Line 34) after the candidate deletion.
4. Otherwise, both gain zero points and must be ranked behind the t 'th position (Line 37) after the candidate deletion.

This justifies the correctness of the ILP and completes the proof for the case of t -Approval.

For the case of t -Veto, it suffices to use the same approach as for t -Approval, provided that we first reverse all preference orders and consider that a candidate is a winner if this candidate's score is the lowest (in essence, this is equivalent to replacing “points” with “vetoes” in the above reasoning). \square

F Brute-Force Search

In this section, we use brute-force search to obtain all of the XP results in this paper and some other FPT results. We note that all these algorithms can be easily adapted to solve the optimization versions of the corresponding problems.

F.1 FPT Results

Let us now focus on Plurality and Veto rules. The main idea for the fixed-parameter tractability results in this section is to guess a subset of voters that will give a specific candidate one point under either Plurality or Veto. The point is that in the case of deleting candidates, after guessing this subset of voters, it is trivial to find the set of candidates to delete to “implement” this guess. This is illustrated in the procedure `CollectCands(\cdot)` given in Algorithm 2.

Lemma 30. *Plurality-CCDC can be solved in $O(m \cdot n \cdot 2^n)$ time, where n is the number of voters and m is the number of candidates in the input election.*

Proof. Let $I := ((C, V), p, k)$ be a Plurality-CCDC instance. If I is a yes-instance, then after deleting at most k candidates, there must be a subset of voters who each give p one point, and no other candidate has more points than p . Observe that in order to let p gain one point from a voter, one has to delete all the candidates this voter prefers to p . Our algorithm, based on these observations, proceeds as follows.

We consider all 2^n subsets of n voters. For each considered set V' of voters we do the following: For each voter v' in V' , we delete all the candidates that v' prefers to p . In effect, all members of V' rank p first. Then, we keep on deleting all the candidates that have more than $\|V'\|$ points (note that deleting some candidate that has more than $\|V'\|$ may result in some other candidate exceeding this bound). If in the end no candidate has more than $\|V'\|$ points and we deleted at most k candidates, we accept. Otherwise, we proceed to the next subset of voters. If we did not accept after going over all subsets of voters, we reject.

To see why the algorithm is correct, note that whenever it accepts, it has just constructed a correct solution. On the other hand, if there is a correct solution in which, after deleting the candidates, p gets points exactly from the voters in some subset V' , then it is easy to see that the algorithm will accept when considering this subset. Establishing the running time is straightforward. \square

It is straightforward to see how to adapt the algorithm from the proof of Lemma 30 to the destructive case. In essence, it suffices to try all choices of a candidate p whose goal is to defeat the despised candidate d and for each such choice guess a subset of voters that are to give points to p . If after deleting the candidates that these voters prefer to p (assuming that neither of them prefers d to p) the despised candidate d has fewer points than p , then we accept. In the destructive case there is no need to have the final loop of deleting candidates scoring higher than p .

Corollary 6. *Plurality-DCDC can be solved in $O(m^2 \cdot n \cdot 2^n)$ time, where n is the number of voters and m is the number of candidates in the input election.*

Lemma 31. *Veto-DCDC can be solved in $O(m \cdot n \cdot 2^n)$ time, where n is the number of voters and m is the number of candidates in the input election.*

Proof. We use almost the same approach as for Lemma 30. First, we guess candidate p whose goal is to have fewer vetoes than d . Deleting candidates can only increase the number of vetoes a remaining candidate has. Thus, our algorithm proceeds as follows.

We consider every subset V' of voters that prefer p to d in the election. For each voter v' in the guessed subset, we delete all the candidates that this voter ranks below d (by choice of V' , p is never deleted). If in effect d has more vetoes than p , we accept. Otherwise we try the next subset of voters. If we do not accept after processing all subsets of voters, we reject.

Verifying the running time and the correctness of this algorithm is immediate. \square

F.2 XP Results

In this section, we establish XP results for all our $W[1]$ -hard problems. This implies that if the number of voters is a constant, the problems are polynomial-time solvable.

Lemma 32. *For each fixed integer t , $t \geq 1$, and for each control type $\mathcal{K} \in \{\text{CCAC}, \text{CCDC}\}$, t -Approval- \mathcal{K} and t -Veto- \mathcal{K} can be solved in time $O^*(m^{tn})$, where m is the number of candidates and n is the number of voters.*

Proof. We consider the CCAC and the CCDC cases jointly, in parallel for both t -Approval and t -Veto. Our algorithm first guesses for each voter the set of t candidates that this voter will rank first (for the case of t -Approval) or last (for the case of t -Veto). There are $O(m^{tn})$ possible different guesses. For each guess, for each voter we verify which candidates have to be added (for the case of CCAC) or deleted (for the case of DCAC) to ensure that the voter ranks the guessed t candidates on top. If it suffices to add/delete k candidates to implement the guess, and in effect of implementing the guess our preferred candidate is a winner, we accept. Otherwise we proceed to the next guess. If no guess leads to acceptance, we reject.

Establishing the correctness and the running time of the algorithm is immediate. \square

Lemma 33. *For each fixed integer t , $t \geq 1$, and each control type $\mathcal{K} \in \{\text{CCAC}, \text{DCAC}\}$, t -Approval- \mathcal{K} and t -Veto- \mathcal{K} can be solved in time $O^*(m^{2tn})$, where m is the number of candidates and n is the number of voters.*

Proof. We use the same approach as described in the proof of Lemma 32, but in addition to guessing the first t candidates for each vote, we also guess for each added candidate c the candidate to whose bundle c belongs. \square

G Miscellaneous Results

Theorem 8. *Maximin-COMB-DCAC is polynomial-time solvable.*

Proof. It was shown by Faliszewski et al. [16] that Maximin-DCAC is polynomial-time solvable. The same strategy can be applied for the combinatorial case as well.

The algorithm is very simple, and can be described as follows: We guess up to two bundles of candidates, add them to the election, and check if the despised candidate d is no longer a winner, if so, we accept and otherwise we reject.

To see why this simple algorithm is correct, consider a solution. If the solution consists of at most two bundles, then we are done. Otherwise, let us take a closer look at the solution. It is clear that in the solution d is not a winner, therefore, there must be at least one other candidate p that has higher score than d . Consider some bundle b_p in the solution which includes p (indeed, there might be several such bundles, and we can choose any one of them arbitrarily; it is also possible that p is present in the original election, in which case we take b_p to be an “empty” bundle). Further, consider some candidate z such that the Maximin score of candidate d in the election E' resulting from adding candidates is exactly $N_{E'}(d, z)$. There may be several such candidates and we choose one arbitrarily. Finally, we choose an arbitrary bundle b_z from the solution that includes z (in fact, it is possible that z is present in the original election, in which case we take b_z to be an “empty” bundle).

It is clear that p defeats d in the solution containing only bundles b_p and b_z (if either of these bundles is “empty”, we simply disregard it). Thus each “yes”-instance of Maximin-DCAC has a solution that consists of at most two bundles and, so, it is enough to guess and test add at most two bundles. \square